UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
MÁSTER EN INGENIERÍA WEB

MIWMessage

Autores: Daniel Lozano Cofrades y David Marchante Cano
Tutor: Luis Fernando de Mingo
10/06/2015

"El reposo no es el destino del hombre,
y la seguridad es sólo una ilusión"

Blaise Pascal

Resumen

El cliente móvil MIWMessaje es una aplicación que permite el intercambio de mensajes de texto entre los usuarios que la utilicen. Esta aplicación es una versión simplificada de aplicaciones como Whatsapp o Hangouts que permiten enviar mensajes firmados y/o cifrados ofreciendo confidencialidad e integridad.

La comunicación entre los clientes se realiza por medio de la plataforma Google App Engine con una aplicación que actúa por medio de un Channel API. En ella está establecido un protocolo https para el intercambio de los mensajes entre los usuarios de la aplicación.

Se utilizarán métodos criptográficos con el fin de proteger la información que viaja por el canal así como la que se almacena en el propio terminal.

Abstract

The mobile client MIWMessage is an application that allows the exchange of the text messages between users that use it. This application is a simplified version of products as Whatsapp or Hangout. With this app, it could be possible to send messages by using cryptography techniques like signed, encrypted or both of them, offering the confidentiality and integrity of the message.

Is carried out by using the Google App Engine, with an application that works through a Channel API. It uses a HTTPS protocol for exchange messages between the users application.

It has been decided to use cryptography methods to achieve the protection of the information that travels through the channel and also to the one that is stored at the device.

Índice de contenidos

Índice de figuras

Índice de tablas

Glosario de términos

- **Texto en claro:** texto sobre el que no se aplica ninguna codificación ni técnica criptográfica, se muestra de forma legible.

- **Criptografía**: (del griego criptos <<oculto>> y grafía <<escritura>>) se define como el ámbito que se ocupa de las técnicas de cifrado o codificado destinadas a alterar las representaciones lingüísticas de ciertos mensajes con el fin de hacerlos ininteligibles a receptores no autorizados. Estas técnicas se utilizan tanto en el Arte como en la Ciencia. Por tanto, el único objetivo de la criptografía era conseguir la confidencialidad de los mensajes. Para ello se diseñaban sistemas de cifrado y códigos.

- **Criptografía simétrica:** (también conocida como "de clave única") es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar. Una vez que ambas partes tienen acceso a esta clave, el remitente cifra un mensaje usando la clave, lo envía al destinatario, y éste lo descifra con la misma clave.

- **Criptografía asimétrica:** (también conocida como "de clave pública") es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que ha enviado el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves.

- **Clave pública:** clave conocida por todos los usuarios. Permite que los usuarios cifren información con esta clave que solo sea descifrable con la clave privada vinculada. También se utiliza para verificar una firma.

- **Clave privada:** clave únicamente conocida por el propio usuario. Permite descifrar mensajes cifrados con la clave pública del propio usuario. También nos permite firmar mensajes.

- **Hash**: (del inglés Resumen) es una función H que es computable mediante un algoritmo,

  $H: U \rightarrow M$

$x \rightarrow h(x)$,

que tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte (mapea) en un rango de salida finito, normalmente cadenas de longitud fija. Es decir, la función actúa como una proyección del conjunto U sobre el conjunto M.

- **Eficiencia:** Capacidad para lograr un fin empleando los mejores medios posibles.
- **Eficacia:** Capacidad para obrar o para conseguir un resultado determinado.
- **PBE:** (acrónimo del inglés Password-Based Encryption) más información aquí.
- **Segundo plano:** en informática, que no se ejecuta en el hilo principal de ejecución, es decir, que continúa ejecutándose aun cuando no es la aplicación que aparece en pantalla.
- **Nonce:** (del inglés: **n**umber used **once** / número usado una sóla vez), es un camino simple para añadir seguridad extra a nuestra aplicación web, evitando particularmente ataques de replay o de reinyección, ataques URL de tipo semánticos y permitiendo verificar además el origen de la petición.
- **Sniffing:** técnica utilizada por un atacante por medio de la cual intercepta los paquetes que viajan por una red con el fin de comprometer su confidencialidad o para realizar ataques maliciosos.
- **Man in the Middle:** es un ataque en el que se adquiere la capacidad de leer, insertar y modificar a voluntad, los mensajes entre dos partes sin que ninguna de ellas conozca que el enlace entre ellos ha sido violado. El atacante debe ser capaz de observar e interceptar mensajes entre las dos víctimas. El ataque MitM es particularmente significativo en el protocolo original de intercambio de claves de Diffie-Hellman, cuando éste se emplea sin autenticación.
- **Java:** lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.
- **API:** (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

1. Introducción

Los smartphones o teléfonos inteligentes se caracterizan por ofrecer una conexión continua inalámbrica a Internet y un hardware con potencia suficiente para realizar tareas complejas. Se han aprovechado estas características por los desarrolladores de software para crear nuevas aplicaciones y usos. De esta manera, este tipo de dispositivos se ha ido haciendo un hueco cada vez mayor en nuestras vidas, habiendo cambiado la forma en la que interaccionamos con nuestro entorno.

A la vez que se crean nuevas aplicaciones y usos para este tipo de dispositivos, también han surgido nuevas amenazas, desde el punto de vista de la seguridad de la información. El uso indiscriminado de redes inalámbricas, la pérdida de datos altamente sensibles, los problemas de privacidad y la aparición de nuevos tipos de software maligno son solo algunas de las amenazas a las que se enfrenta este nuevo paradigma de computación. Es por ello, que se hace necesario aprender a desarrollar sistemas y aplicaciones que tengan en cuenta estas amenazas y que tengan en cuenta la seguridad de la información como uno de sus principios fundamentales.

En el presente Trabajo Fin de Máster se propone desarrollar una aplicación para el sistema operativo Android que permita el envío de mensajes entre los diferentes usuarios de la misma. Para su desarrollo se hará especial énfasis en la seguridad de los datos transmitidos y almacenados en el dispositivo móvil.

1.1 Objetivos

El principal propósito de este proyecto es desarrollar una aplicación móvil de comunicación entre diferentes terminales. A continuación se especifican los objetivos principales:

1. Desarrollar la aplicación Android que permita el envío de mensajes entre los usuarios de la aplicación.

2. Aplicar métodos criptográficos para proteger la información sensible que viaja por el canal.

Estos objetivos, poseen también estos sub-objetivos:

- Ampliar conocimientos sobre el desarrollo de aplicaciones Android

- Estudiar la plataforma Google Engine.
- Analizar la comunicación con el Channel API
- Integrar el Sistema implementado.
- Evaluar el Sistema desarrollado

1.2 Fases de desarrollo

Dentro del Proyecto se incluyen una serie de fases que detallamos a continuación:

Definición

Esta fase enmarca una descripción del objeto de este proyecto, así como los recursos utilizados para llevarlo a cabo.

Análisis, Diseño e Implementación

En esta fase se va a enmarcar la solución aportada para el planteamiento inicial. En ella se detallan las características esenciales del desarrollo.

Evaluación

En este paso se realizará una evaluación para validar la solución implementada, mostrando que la solución cumple con los requisitos definidos en la sección anterior mediante un plan de pruebas.

1.3 Recursos empleados

Los recursos utilizados para desarrollar la solución obtenida se pueden dividir en dos categorías principales: fuentes bibliográficas para la comprensión y el uso de plataformas de comunicación y mensajería; y las herramientas de desarrollo y documentación necesarios para el desarrollo del proyecto.

- Google Scholar.
- Biblioteca de la Escuela Técnica Superior de Ingeniería. Universidad Politécnica de Madrid.

Las herramientas de desarrollo y documentación utilizada han sido:

- Herramientas:
  - Google Engine Channel API
  - Android 4.4.2
- Documentation:
  - Google App Engine
  - Android API (Android Developers)
  - Channel API Google Cloud Platform

1.4 Estructura del documento

Este documento se divide en las siguientes secciones:

- **Primera sección**: tiene por objeto introducir el trabajo realizado y el contenido del documento.

- **Segunda sección:** tiene como objeto principal la gestión de proyectos software. Se refleja el alcance del proyecto, plan de trabajo, gestión de recursos, gestión de riesgos y plan de pruebas.

- **Tercera sección**: acoge lo relativo a la implementación que se ha aplicado al planteamiento inicial; es decir, recoge las fases de desarrollo e implementación así como los resultados obtenidos.

2 Gestión de proyecto software

El objetivo de este apartado es realizar una simulación de la gestión del proyecto el software desarrollado. Consiste en una estimación hipotética del proyecto como si fuese a desarrollar con las condiciones habituales de un entorno empresarial. El proyecto ha tenido una duración de 2 meses, desde Mayo hasta Junio de 2015, ambos incluidos. En los siguientes apartados se incluye la estimación de las tareas y recursos empleados así como el presupuesto que se estima para desarrollar dicho proyecto.

2.1 Estimación de tareas y recursos

En este apartado se presentan los costes asociados al desarrollo del proyecto. Se tienen en cuenta los costes de personal, costes de hardware y de software.

Costes de personal

Los costes de personal los constituyen los salarios del equipo encargado de desarrollar el proyecto. En primer lugar, se definirán los roles de los empleados:

- **Analista**: es la persona que trabaja con el cliente para realizar el análisis y especificación del sistema, es decir, dividir el problema a resolver en sub-problemas de menos complejidad.

- **Programador**: es la persona encargada de convertir la especificación del sistema en código fuente ejecutable utilizando uno o más lenguajes de programación, así como herramientas de software de apoyo a la programación.

El equipo de desarrollo del proyecto contará con un empleado por cada uno de los roles definidos anteriormente. El salario dependerá del rol que desempeñe el empleado así como la jornada de trabajo. Desempeñaran una jornada de 2h diarias durante la duración del proyecto. El salario bruto mensual se calculará de la siguiente forma:

$$Salario\ bruto\ mensual = \frac{coste}{hora} \times \frac{horas}{día} \times \frac{días}{mes}$$

A continuación se mostrará la tabla salarial del equipo de trabajo:

Empleado	coste/hora (€)	jornada(h)	dias/mes	salario bruto mensual (€)
Analista	13	2	15	390
Programador	10	2	21	420
TOTAL (€)				810

Tabla 1: Salario bruto mensual del equipo de trabajo

A continuación se muestran los costes asociados a los dos meses del proyecto, teniendo en cuenta la cuota de la seguridad social, cuyo porcentaje es del 23,6% según las bases y tipos de cotización [BTC] establecido por el Ministerio de Empleo y Seguridad Social. Se calculará el coste total de cada empleado en el proyecto siguiendo esta fórmula:

$$Coste\ total = (salario\ bruto\ mensual + cuota\ SS) \times meses$$

A continuación se mostrarán los costes totales de personal:

Empleado	Salario bruto mensual (€)	Cuota Seguridad Social	Meses de trabajo	Coste total (€)
Analista	390	92,04	2	964,08
Programador	420	99,12	2	1038,24
TOTAL (€)				2002,32

Tabla 2: Costes totales de personal

2.2 Presupuesto

Una vez realizado el desglose de los costes de personal, si nos centramos en los gastos de software y hardware no sería necesario incluir un gasto representativo ya que todo el software utilizado es de licencia gratuita o proporcionada por la propia universidad; y el hardware es también el propio de la universidad por lo que el coste total se limita a los costes de personal. El coste asociado al proyecto es el que se muestra en la siguiente tabla:

Concepto	Coste (€)
Costes de personal	2002,32
Costes hardware	0
Costes software	0
Total (€)	2002,32

Tabla 3: Costes totales

Para concluir, se presenta en la tabla 3, el presupuesto final incluyendo riesgos, beneficios e IVA:

Concepto	Coste (€)
Costes totales	2.002,32 €
Riesgos (15%)	300,35 €
Beneficios (20%)	400,46 €
TOTAL (sin IVA)	2.703,13 €
IVA (21%)	567,66 €
Total (€)	3.270,79 €

Tabla 4: Presupuesto final

El presupuesto total del proyecto expresado en el presente documento asciende a 3.270,79€ (TRES MIL DOSCIENTOS SETENTA EUROS CON SETENTA Y NUEVE CENTIMOS)

Madrid a 22 de Junio de 2015

El ingeniero proyectista

2.3 Plan de trabajo

En este apartado se especifica el plan de trabajo que se ha determinado para el desarrollo del proyecto. Se expondrán la clasificación de tareas así como su planificación temporal a lo largo del recorrido del proyecto.

Identificación de tareas

A continuación se especifican las tareas que componen el proyecto:

- **Análisis del problema:** se describen las principales características del problema a resolver por medio de la identificación de **requisitos**.

- **Modelado de datos:** se realizará un estudio de los datos que requiere almacenar la aplicación cliente en local.

- **Diseño de la interfaz:** se define la interfaz del sistema, especificando todos los elementos (transiciones, interfaces, componentes) que se desarrollan en el entorno.

- **Implementación. Fase 1:** esta fase hace referencia a la implementación del cliente de mensajería. Se desarrollan toda la aplicación para que permita el tratamiento de usuarios y el intercambio de mensajes en función del canal de comunicación definido por el servidor (sin persistencia).

- **Implementación. Fase 2:** esta fase hace referencia a la implementación del modelo de datos en el cliente de mensajería. Se crea una base de datos local en la aplicación cliente para almacenar toda la información que el cliente necesita para permitir que la aplicación siga conservando toda la información una vez cerrada la sesión.

- **Implementación. Fase 3:** esta fase hace referencia a la integración de los métodos criptográficos. Se ampliará el resultado de la Fase 1, permitiendo el cifrado y firmado de los mensajes que viajan.

- **Implementación. Fase 4:** esta fase hace referencia a la protección de la información sensible que se almacena en el cliente. Se ampliará el resultado de la Fase 2, permitiendo el guardado de la información sensible en la base de datos mediante métodos criptográficos de cifrado.

- **Pruebas:** esta fase corresponde con las pruebas que se realizan sobre la aplicación siguiendo el plan de pruebas especificado.

- **Redacción de memoria:** en esta fase se elabora el presente documento, correspondiente a la documentación del proyecto completo.

2.4 Gestión de riesgos

En este apartado, se expondrán y analizarán los diferentes riesgos ligados al desarrollo del trabajo.

Identificación de riesgos

A continuación se especificarán los riesgos asociados a este proyecto, siendo estos, relevantes para el desarrollo de este proyecto concreto:

- Recursos no disponibles
- Planificación demasiado optimista
- Problema en la construcción de la aplicación
- Estabilidad de la interfaz
- Diseño inadecuado (hay que realizar un nuevo diseño)
- Análisis inadecuado de las capacidades del software (hay que estudiar ampliar funcionalidad de forma externa.
- Documentación insuficiente

Análisis de riesgos

En este apartado analizaremos la probabilidad y la magnitud de la pérdida en semanas en el caso sé que ocurriera cada uno de los riesgos especificados en el apartado anterior:

Riesgo	Probabilidad	Magnitud de perdida (semanas)	Exposición a riesgo (semanas)
Recursos no disponibles	10%	1	0,1
Planificación demasiado optimista	10%	4	0,4
Problema en la construcción de la aplicación	15%	4	0,6
Estabilidad de la interfaz	5%	2	0,1
Diseño inadecuado (hay que realizar un nuevo diseño)	10%	2	0,2
Análisis inadecuado de las capacidades del software (hay que estudiar ampliar la funcionalidad de forma externa)	20%	3	0,6
Documentación insuficiente	10%	1	0,1

Tabla 5: Análisis de riesgos

3 Análisis, Diseño e Implementación

El objetivo de este apartado consiste en presentar de qué forma se abordan los objetivos principales del trabajo presentados en el apartado 1.1 Objetivos. En un primer apartado se contempla el análisis de la aplicación que se implementará, que acotaría el "qué se va a hacer" y el diseño, en el que se define el "cómo se va a hacer".

Se realizará una aplicación alojada en un cliente Android en la que el usuario pueda comunicarse con otros usuarios por medio de un canal seguro, así como las tareas referentes a organización y sistema de autentificación de usuarios.

A continuación se indica cómo se han cubierto los objetivos expuestos en el apartado objetivos:

- La aplicación debe **permitir el envío de mensajes entre los usuarios de la aplicación.** Esto se cubre de dos formas distintas. Primero por medio de un sistema de autenticación contra el servidor que permita establecer una canal de comunicación para cada uno de los usuarios potenciales de la aplicación. En segundo lugar, la aplicación permite acceder a los contactos almacenados en el teléfono y verificar cuáles de ellos están registrados en la aplicación, permitiendo así establecer comunicación con ellos.

- La aplicación **deberá Aplicar métodos criptográficos para proteger la información sensible que viaja por el canal**. Para transmitir la información de forma segura aplicando técnicas de cifrado y firmado; consiguiendo así, verificar el origen del mensaje y proteger su contenido.

3.1 El proceso de desarrollo

En este apartado se detalla el proceso de desarrollo que se ha llevado a cabo para realizar el proyecto, detallando el modelo del proceso aplicado para elaborar la solución y las fases de las que consta la misma. El desarrollo consta de cuatro fases fundamentales: la **fase de análisis**, donde se detallan los requisitos del sistema; la **fase de diseño**, donde se expone la arquitectura del sistema, su organización y las interfaces de las que constará la aplicación; y la **fase de pruebas** donde se detallan las pruebas realizadas sobre la aplicación.

Modelo del proceso

El ciclo de vida de un proyecto software se definen como un marco de referencia en el que se incluyen los procesos, actividades y tareas que forman parte del proceso de desarrollo, mantenimiento y despliegue de un proyecto software. El modelo de ciclo de vida lo componen diferentes fases expresadas en la introducción de este apartado. El modelo de proceso empleado para elaborar la solución ha sido el **modelo en cascada** [CAS], en el que se realiza un desarrollo vertical y las fases se van apoyando en fases anteriores.

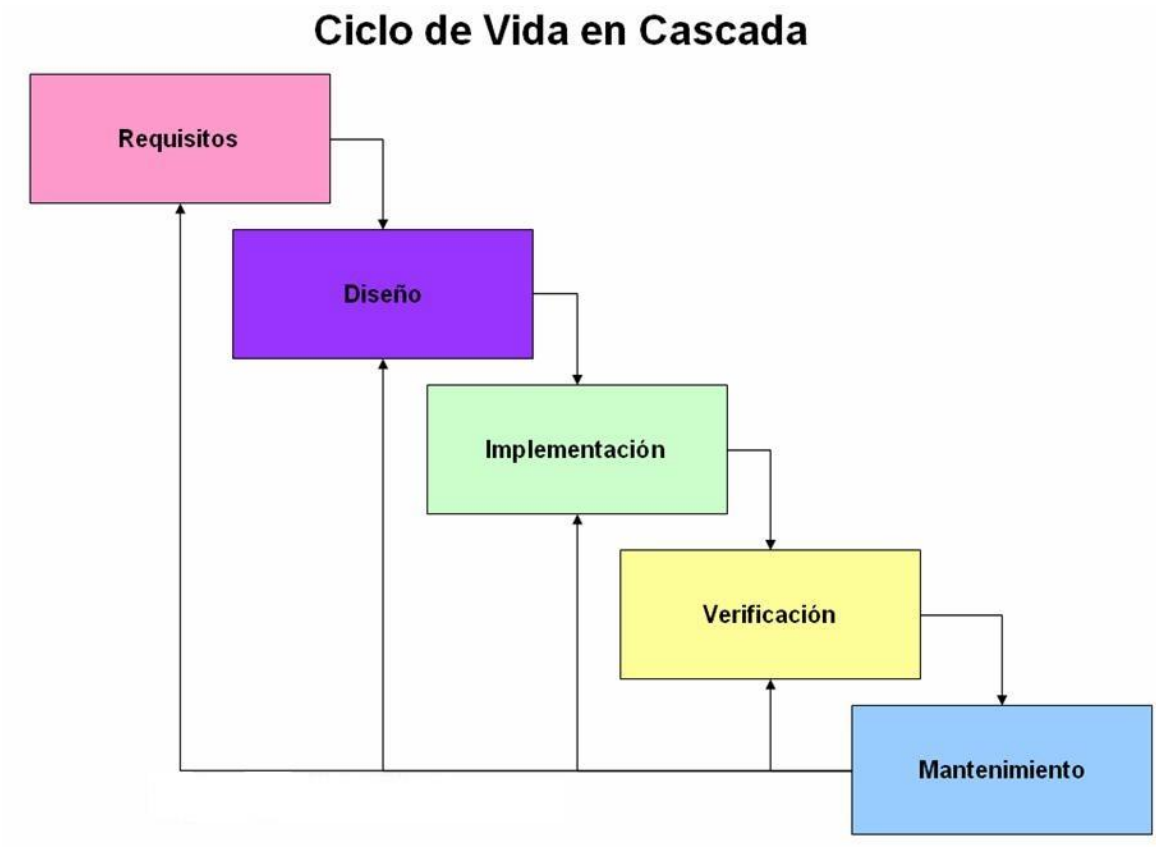

Figura 1: Ciclo de vida del modelo en cascada

Análisis

La fase análisis está dirigida a estudiar las características del proyecto con el propósito desarrollar una solución que lo resuelva. Para definir este listado de requisitos, se utilizará la plantilla de Volere [VOL]. Dicha plantilla ha sido adaptada según las necesidades del sistema es siendo estos, los campos relevantes para el desarrollo de este proyecto:

- **Identificador de requisito (RX-NN):** descriptor que caracteriza unívocamente a un requisito. **R** representa que es un requisito, **X** representa el tipo de requisito (de carácter funcional o no funcional) y **NN** representa el número del requisito.
- **Nombre:** descriptor inicial del requisito.
- **Fuente:** origen del requisito.
- **Descripción:** descripción del requisito.
- **Claridad:** mide la ambigüedad de cada uno de los requisitos. Los valores pueden ser alta, media y baja; siendo de claridad alta un requisito con alto nivel de ambigüedad, y baja, un requisito con bajo nivel de ambigüedad.
- **Prioridad:** define el nivel de importancia del requisito. La prioridad puede ser alta, media o baja, dependiendo del grado de relación que tenga dicho requisito respecto a los objetivos del proyecto. Es decir, los requisitos tendrán una prioridad alta si están fuertemente ligados a los objetivos del proyecto, y tendrán una prioridad baja si tienen una leve relación con los objetivos.

A continuación se muestra la plantilla empleada para realizar la especificación de requisitos:

RX – XX	
Nombre	
Fuente	
Prioridad	
Claridad	
Descripción	

Tabla 6: Adaptación de la plantilla de requisitos de Volere

Definición de requisitos

A continuación se procederá a enumerar todos los requisitos del sistema clasificados entre funcionales y no funcionales. Identificaremos los requisitos según el identificador de requisito definido en el apartado anterior

Requisitos funcionales

Los requisitos funcionales representan las declaraciones de los servicios que debe proporcionar el sistema, es decir, representan las capacidades a cubrir por el sistema.

RF – 01			
Nombre	Registrar		
Fuente	Cliente		
Prioridad	Alta	Claridad	Alta
Descripción	Se le permitirá al usuario registrarse en el sistema		

Tabla 7: Requisito funcional RF-01

RF – 02			
Nombre	Acceder		
Fuente	Cliente		
Prioridad	Alta	Claridad	Alta
Descripción	Se le permitirá al usuario acceder estando registrado		

Tabla 8: Requisito funcional RF-02

RF – 03			
Nombre	Cambiar usuario		
Fuente	Cliente		
Prioridad	Media	Claridad	Alta
Descripción	Se le permitirá al usuario cambiar de usuario por otro previamente registrado en el mismo terminal		

Tabla 9: Requisito funcional RF-03

RF – 04			
Nombre	Solicitar contactos		
Fuente	Cliente		
Prioridad	Media	Claridad	Alta
Descripción	Se pedirán los contactos de la agenda del teléfono que estén registrados.		

Tabla 10: Requisito funcional RF-04

RF – 05			
Nombre	Recibir mensajes al conectar		
Fuente	Cliente		
Prioridad	Media	Claridad	Alta
Descripción	Recibirá los mensajes que le hayan enviado al usuario mientras que la aplicación no estaba activa.		

Tabla 11: Requisito funcional RF-05

RF – 06			
Nombre	Enviar mensaje en claro		
Fuente	Cliente		
Prioridad	Alta	Claridad	Alta
Descripción	Se le permitirá al usuario enviar mensajes con el texto en claro.		

Tabla 12: Requisito funcional RF-06

RF – 07			
Nombre	Enviar mensaje cifrado		
Fuente	Cliente		
Prioridad	Alta	Claridad	Media
Descripción	Se le permitirá al usuario enviar mensajes cifrados.		

Tabla 13: Requisito funcional RF-07

RF – 08			
Nombre	Enviar mensaje cifrado		
Fuente	Cliente		
Prioridad	Alta	Claridad	Media
Descripción	Se le permitirá al usuario enviar mensajes firmados		

Tabla 14: Requisito funcional RF-08

Requisitos no funcionales

Los requisitos no funcionales representan la forma de los servicios que debe proporcionar el sistema, representan servicios y funciones ofrecidas por el sistema.

RNF – 01			
Nombre	Idioma interfaz		
Fuente	Cliente		
Prioridad	Alta	Claridad	Alta
Descripción	La totalidad de la interfaz se mostrará en castellano.		

Tabla 15: Requisito no funcional RFN-01

RNF – 02			
Nombre	Compatibilidad		
Fuente	Cliente		
Prioridad	Alta	Claridad	Alta
Descripción	La aplicación será compatible con versiones de Android superiores a la 4.4.2		

Tabla 16: Requisito no funcional RNF-02

RNF – 03			
Nombre	Sencillez en el interfaz		
Fuente	Cliente		
Prioridad	Alta	Claridad	Media
Descripción	Se utilizará una interfaz sencilla.		

Tabla 17: Requisito no funcional RNF-03

RNF – 04			
Nombre	Interfaz en columnas		
Fuente	Cliente		
Prioridad	Alta	Claridad	Alta
Descripción	el interfaz principal contará con los tres modos de autenticación: Registro, cambio de usuario y conexión por ultimo usuario conectado.		

Tabla 18: Requisito no funcional RNF-04

RNF – 05			
Nombre	Duración mensajes de aviso		
Fuente	Cliente		
Prioridad	Baja	Claridad	Alta
Descripción	Los mensajes de aviso se mostrarán un tiempo superior a 2 segundos.		

Tabla 19: Requisito no funcional RNF-05

RNF – 06			
Nombre	Visualización		
Fuente	Cliente		
Prioridad	Media	Claridad	Media
Descripción	El entorno será visualizado un en una pantalla de mínimo 4.3 pulgadas.		

Tabla 20: Requisito no funcional RNF-06

RNF – 07			
Nombre	Pestañas de chat		
Fuente	Cliente		
Prioridad	Media	Claridad	Alta
Descripción	Se mostrarán 2 pestañas en la interfaz de chat: preferencias y contactos.		

Tabla 21: Requisito no funcional RNF-06

RNF – 08			
Nombre	Aplicación en segundo plano		
Fuente	Cliente		
Prioridad	media	Claridad	Alta
Descripción	La aplicación se mantendrá abierta en segundo plano.		

Tabla 22: Requisito no funcional RNF-08

RNF – 09			
Nombre	Envío POST		
Fuente	Cliente		
Prioridad	Alta	Claridad	Alta
Descripción	Los mensajes se enviarán por método POST.		

Tabla 23: Requisito no funcional RNF-09

RNF – 10			
Nombre	Conexión HTTPS		
Fuente	Cliente		
Prioridad	Media	Claridad	Alta
Descripción	Se conectará por medio del protocolo seguro HTTPS.		

Tabla 24: Requisito no funcional RNF-10

RNF – 11			
Nombre	Logo corporativo		
Fuente	Cliente		
Prioridad	Media	Claridad	Media
Descripción	El logo de la aplicación poseerá los colores representativos del Master en Ingeniería Web (azul y naranja).		

Tabla 25: Requisito funcional RNF-11

Casos de Uso

En esta fase se presentan los casos de uso obtenidos a partir de la definición de requisitos en las páginas anteriores. De manera general, un caso de uso no es más que una descripción de los pasos que el usuario debe realizar para llevar a cabo un proceso. En este caso concreto, sirven para definir los pasos que tiene que llevar a cabo un usuario para cumplir la funcionalidad descrita en los requisitos.

A continuación se mostrará el Diagrama de Casos de Uso en el que contamos con 2 actores: usuario registrado y usuario no registrado.

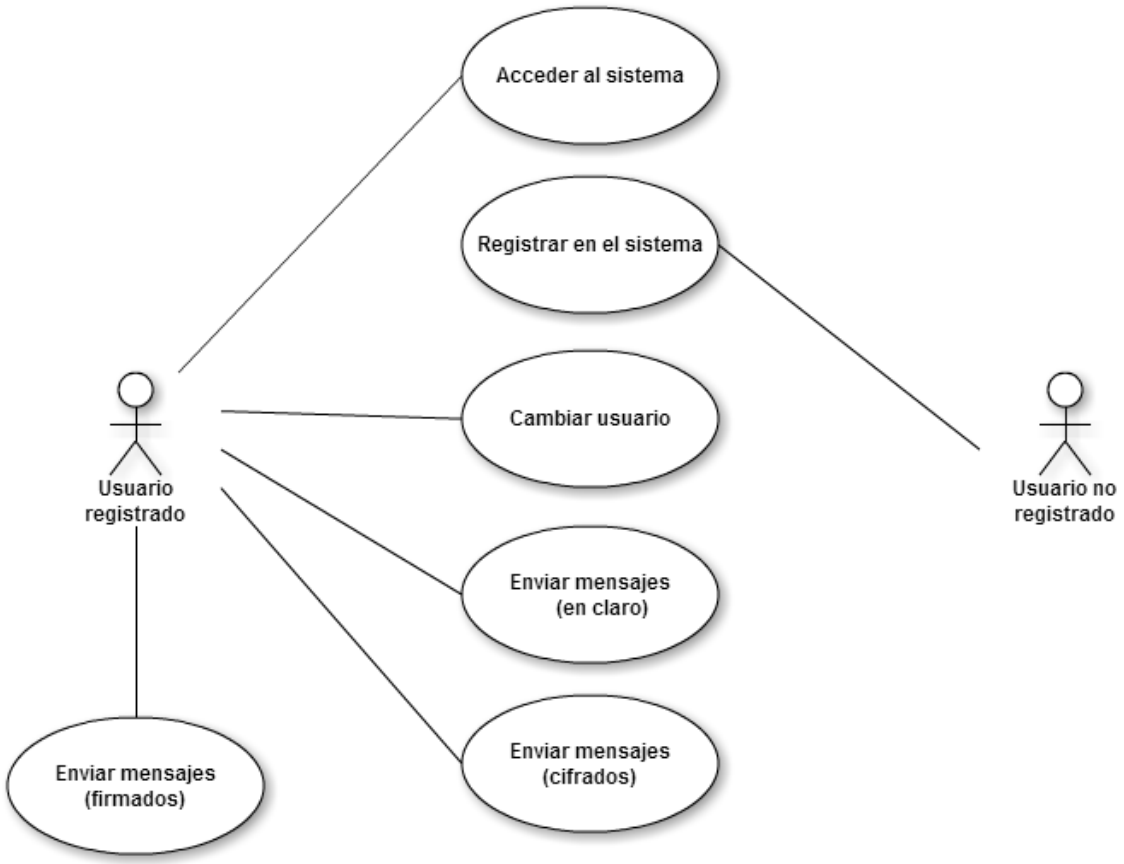

Figura 2: Diagrama de Casos de Uso

A continuación se presentan los campos relevantes para la correcta identificación de los casos de uso:

- **Identificador de requisito (CU-NN):** escritor que caracteriza unívoca mente a un caso de uso. **CU** representa que es un Caso de Uso y **NN** representa el número del Caso de Uso.
- **Nombre:** descriptor inicial del Caso de Uso.
- **Actores:** Entidad externa al sistema que demanda la funcionalidad dada por el caso de uso**.**
- **Escenario:** Explica en qué contexto se aplica el caso de uso.
- **Precondiciones:** Condiciones que se deben cumplir para que el curso de eventos pueda llevarse a cabo.
- **Curso básico:** Especifica la interacción entre los actores y el sistema.

A continuación se presentan los Casos de Uso extraídos de los requisitos:

CU-01	
Nombre	Acceder al sistema
Actores	Usuario registrado
Descripción	El usuario desea acceder a la aplicación
Precondiciones	El usuario debe tener una cuenta en la aplicación
Curso básico	1. El usuario indica que ya está registrado en su terminal. 2. El sistema pide que introduzca la contraseña. 3. El usuario accede a la aplicación.

Tabla 26: Caso de Uso CU-01

CU-02	
Nombre	Registrar en el sistema
Actores	Usuario no registrado
Descripción	El usuario desea registrarse en la aplicación
Precondiciones	-
Curso básico	1. El usuario introduce el número a registrar. 2. El usuario indica que quiere registrarse. 3. El usuario introduce el resto de datos de registro. 4. El usuario se registra en la aplicación.

Tabla 27: Caso de Uso CU-02

CU-03	
Nombre	Cambiar de usuario
Actores	Usuario registrado
Descripción	El usuario desea cambiar el usuario de la aplicación
Precondiciones	El usuario debe tener una cuenta en la aplicación
Curso básico	1. El usuario introduce el número de usuario al que desea cambiar. 2. El usuario indica que quiere cambiar de usuario. 3. El usuario introduce la contraseña. 4. Se cambia el usuario en la aplicación.

Tabla 28: Caso de Uso CU-03

CU-04	
Nombre	Enviar mensajes en claro
Actores	Usuario registrado
Descripción	El usuario desea enviar un mensaje en claro
Precondiciones	1. Tener deshabilitadas las opciones criptográficas 2. El usuario debe haber iniciado una conversación con un contacto.
Curso básico	1. El usuario introduce el mensaje que desea enviar. 2. El usuario indica que quiere enviar el mensaje.

Tabla 29: Caso de Uso CU-04

CU-05	
Nombre	Enviar mensajes cifrados
Actores	Usuario registrado
Descripción	El usuario desea enviar un mensaje cifrado
Precondiciones	1. Habilitar la opción de cifrado. 2. El usuario debe haber iniciado una conversación con un contacto.
Curso básico	1. El usuario introduce el mensaje que desea enviar. 2. El usuario indica que quiere enviar el mensaje.

Tabla 30: Caso de Uso CU-05

CU-06	
Nombre	Enviar mensajes firmados
Actores	Usuario registrado
Descripción	El usuario desea enviar un mensaje firmado
Precondiciones	1. Habilitar la opción de firmado. 2. El usuario debe haber iniciado una conversación con un contacto.
Curso básico	1. El usuario introduce el mensaje que desea enviar. 2. El usuario indica que quiere enviar el mensaje.

Tabla 31: Caso de Uso CU-06

A continuación, se muestran las matrices de trazabilidad que relacionan cada uno de los requisitos presentados en el apartado definición de requisitos con los casos de uso que se han presentado en el presente apartado. Se presentarán dos matrices en función de la naturaleza de los requisitos que se trazan con los casos de uso.

	CU-01	CU-02	CU-03	CU-04	CU-05	CU-06
RF-01		X				
RF-02	X					
RF-03			X			
RF-04	X		X	X	X	X
RF-05				X	X	X
RF-06				X		
RF-07					X	
RF-08						X

Tabla 32: Matriz de trazabilidad: Requisitos funcionales vs. Casos de Uso

	CU-01	CU-02	CU-03	CU-04	CU-05	CU-06
RNF-01	X	X	X	X	X	X
RNF-02						
RNF-03	X	X	X	X	X	X
RNF-04	X	X	X			
RNF-05	X	X	X	X	X	X
RNF-06				X	X	X
RNF-07						
RNF-08						
RNF-09	X	X	X	X	X	X
RNF-10	X	X	X	X	X	X
RNF-11						

Tabla 33: Matriz de trazabilidad: Requisitos no funcionales vs. Casos de Uso

Diseño

El entorno desarrollado se trata de un modelo de aplicación distribuida en la que las tareas se reparten entre servidor y cliente. El servidor es el encargado de proveer los recursos y servicios que son demandados por el cliente.

La aplicación cliente se ha desarrollado en Android (versión 4.4.2) sobre el entorno eclipse. El motivo de utilizar estas plataforma ha sido principalmente por dos razones: la posibilidad de ampliar los conocimientos adquiridos en el Master en Ingeniería Web para clientes móviles Android; y montarlo sobre un entorno conocido y de fácil integración con la tecnología cliente como es Eclipse.

Esta fase la subdividiremos en los siguientes apartados: modelo de datos, interfaces de la aplicación cliente, comunicación cliente-servidor y seguridad.

Modelo de datos

A continuación se definirá la estructura de la base de datos que poseerá la aplicación cliente. Se mostrará tanto el modelo de entidad relación como los campos de las entidades que se encargarán de la persistencia de la aplicación.

Modelo entidad-relación

Cómo se puede apreciar en el modelo de entidad-relación, se cuenta con tres tablas que son: USUARIO, CONTACTO y MENSAJE; todas relacionadas entre sí con una relación 1:N como se puede apreciar en la siguiente figura.

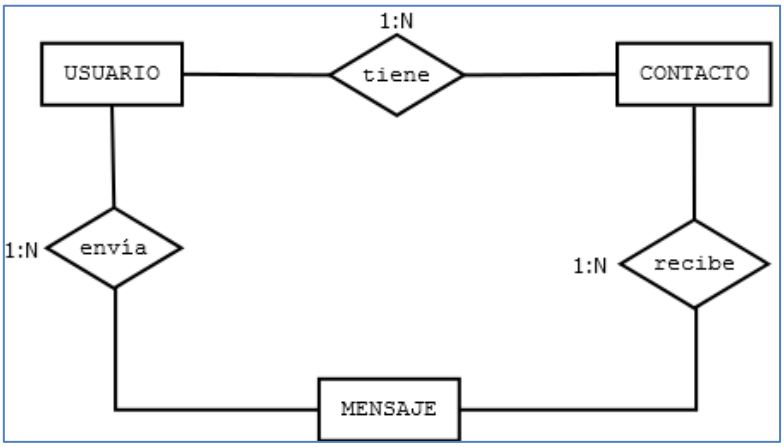

Figura 3: Modelo Entidad-Relación

Entidades del modelo de datos

En esta sección se va a mostrar cada tabla de la base de datos indicando qué campos de cada una de ellas han sido cifrados y por qué. A continuación se mostrará cada tabla seguida de su explicación.

Tabla USUARIO

En esta tabla se almacena la información de cada uno de los usuarios de la aplicación.

TABLA	CAMPOS	TIPO	DESCRIPCIÓN
USUARIO	TOKEN	Text	Token del usuario
	NICK	Text	Nick del usuario
	MOBILE	Text	Número de móvil del usuario
	PASSWORD	Text	Contraseña del usuario
	SALT	Text	Nonce para realizar el password Based Encryption del usuario
	CLAVEPUBLICA	Text	Clave pública del usuario
	CLAVEPRIVADA	Text	Clave privada del usuario
	FECHACLAVES	Text	Fecha de generación de las claves RSA del usuario

Tabla 34: Tabla Usuario

Tabla CONTACTO

En esta tabla se almacena la información de cada uno de los contactos de un usuario de la aplicación.

TABLA	CAMPOS	TIPO	DESCRIPCIÓN
CONTACTO	NUMEROCONTACTO	Text	Número de móvil del contacto
	NICK	Text	Nick del contacto
	TOKEN	Text	Token del usuario que posee ese contacto
	CLAVEPUBLICA	Text	Clave pública del contacto
	CONTACTOAGENDA	Integer	Entero para saber si el contacto es de la agenda o no para que a la hora de actualizar no los borremos porque entonces no los volveremos a cargar

Tabla 35: Tabla Contacto

Tabla MENSAJE

En esta tabla se almacena la información de cada uno de los mensajes de un usuario con un contacto de la aplicación.

TABLA	CAMPOS	TIPO	DESCRIPCIÓN
MENSAJE	NUMEROCONTACTO	Text	Número de móvil del contacto
	TOKEN	Text	Token del usuario de la aplicación
	MENSAJE_CHAT	Text	Mensaje de chat
	RECIBIDO_ENVIADO	Integer	Flag que indica si el mensaje es enviado o recibido
	CIFRADO	Integer	Flag que indica si el mensaje está cifrado
	IDMENSAJESECUENCIAL	Integer	Entero secuencial para mostrar los mensajes en orden
	HORAMENSAJE	Text	Hora del mensaje
	FECHAMENSAJE	Text	Fecha del mensaje
	FIRMADO	Integer	Flag que indica si el mensaje está firmado
	VERIFICADO	Integer	Flag que indica si el mensaje ha sido verificado

Tabla 36: Tabla Mensaje

Diseño de los Interfaces

En este apartado, se detallan cada una de las interfaces utilizadas en la aplicación.

Pantalla Principal (main)

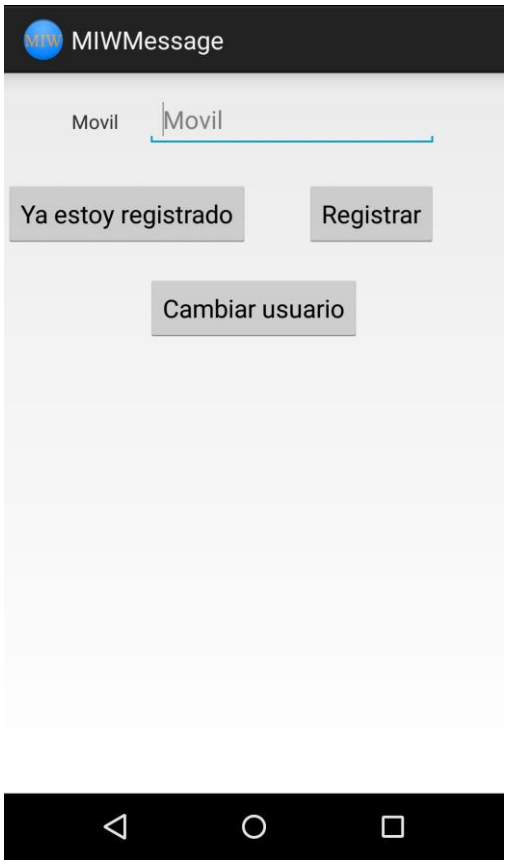

Figura 4: Pantalla principal

En la pantalla principal, se contemplan tres formas diferentes de entrar en la aplicación: por medio de **registro**, **cambio a un usuario** previamente almacenado en la BBDD y registro **según el último acceso**.

Pantalla Logueo

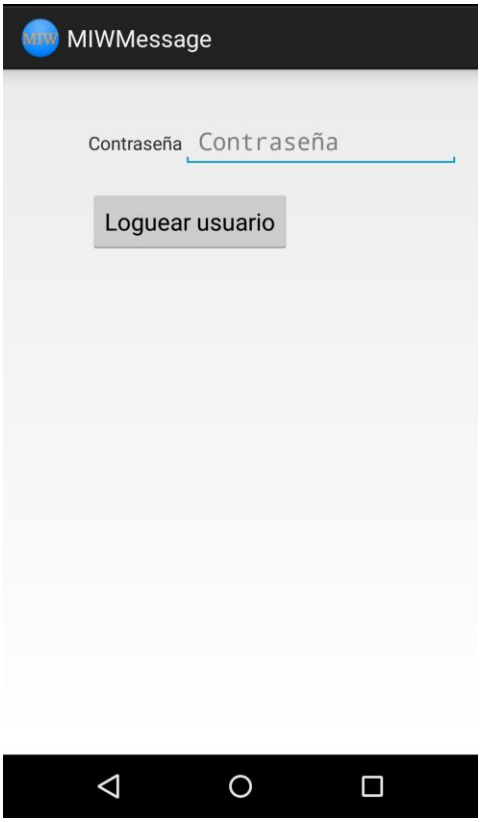

Figura 5: Pantalla Logueo

En la pantalla logueo permite introducir la contraseña del usuario con el que se desea acceder para autenticarse en el sistema.

Pantalla Registro

Figura 6: Pantalla Registro

En la pantalla registro se introducirá el Nick con el que se desea registrarse en la aplicación y la contraseña, repitiendo ésta para asegurar que no se cometa ningún error al ser introducida.

Pantalla Contactos

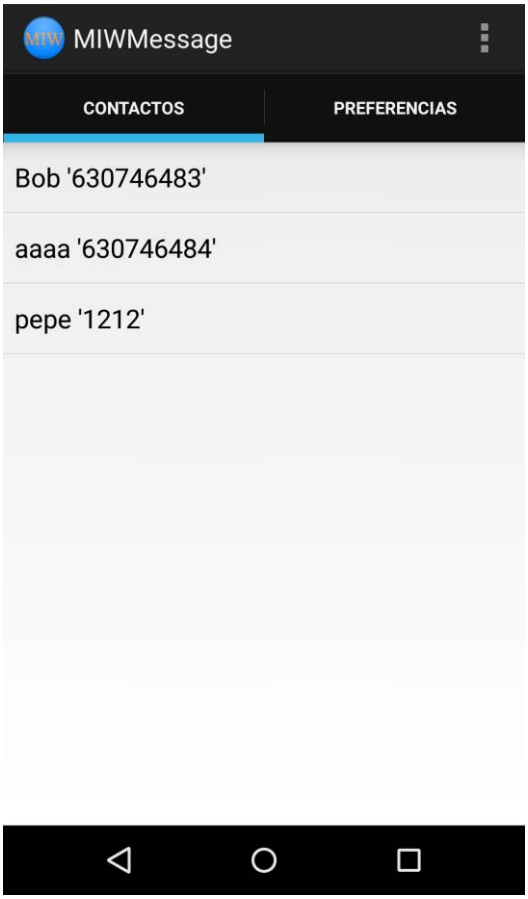

Figura 7: Pantalla Contactos

En la pantalla de contactos se visualizan los contactos de la agenda del usuario que están registrados en la aplicación.

Pantalla Preferencias

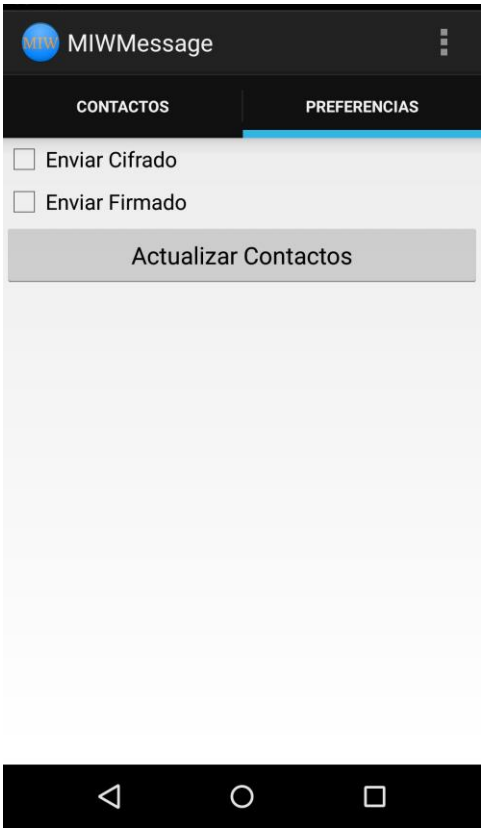

Figura 8: Pantalla Preferencias

En esta interfaz, se contemplan las diferentes técnicas criptográficas que se le pueden aplicar al mensaje: por medio **cifrado**, **firmado** o **ambas**. Aparte, da la opción de actualizar los contactos registrados de MIWMessage.

Pantalla Chat

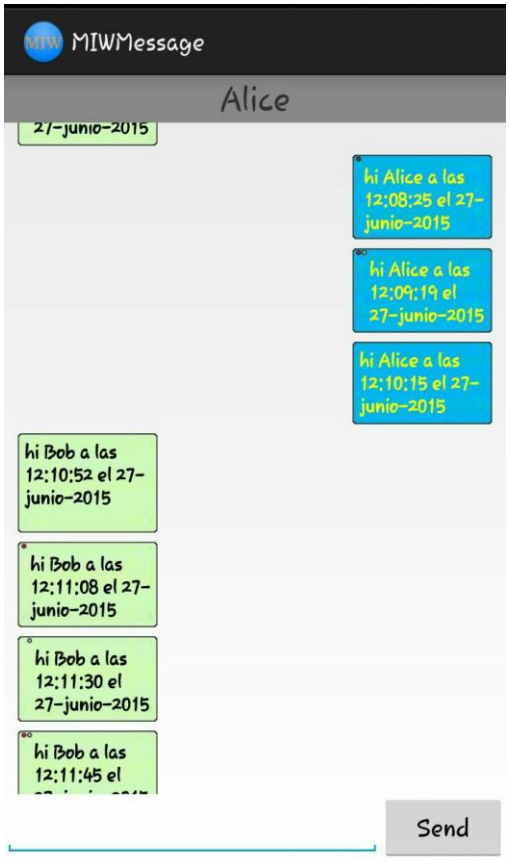

Figura 9: Pantalla Chat

En la pantalla de chat se mostrarán los mensajes que se vayan intercambiando con un contacto. En la parte superior aparece el nombre del contacto con el que se está manteniendo la conversación.

A continuación se muestra la información visual que reflejará la apariencia de las burbujas de la conversación del chat

Burbujas de conversación

Se han definido una serie de identificadores para conocer las herramientas criptográficas aplicadas a cada uno de los mensajes. Para esto, se aplican estilos a las burbujas de conversación como se define a continuación:

El color de la burbuja de mensajes enviados (azul) es distinto al color de los mensajes recibidos (verde). Además dentro de cada burbuja puede haber dos pequeños círculos antes del mensaje que indican si el mensaje está firmado, firmado no verificado, cifrado, cifrado y firmado o ninguna de las anteriores si no aparecen dichos círculos. La leyenda para entender los círculos es la siguiente:

- Firmado: círculo gris
- Cifrado: círculo rojo
- Cifrado y firmado: ambos círculos (gris y rojo).
- Firmado no verificado: círculo gris tachado con una X.
- Sin firmar y Sin cifrar: ningún círculo.

Mapa de pantallas

A continuación se muestra el mapa de pantallas. En él, se define la navegabilidad entre cada una de las interfaces de la aplicación en función de las acciones que realice el usuario.

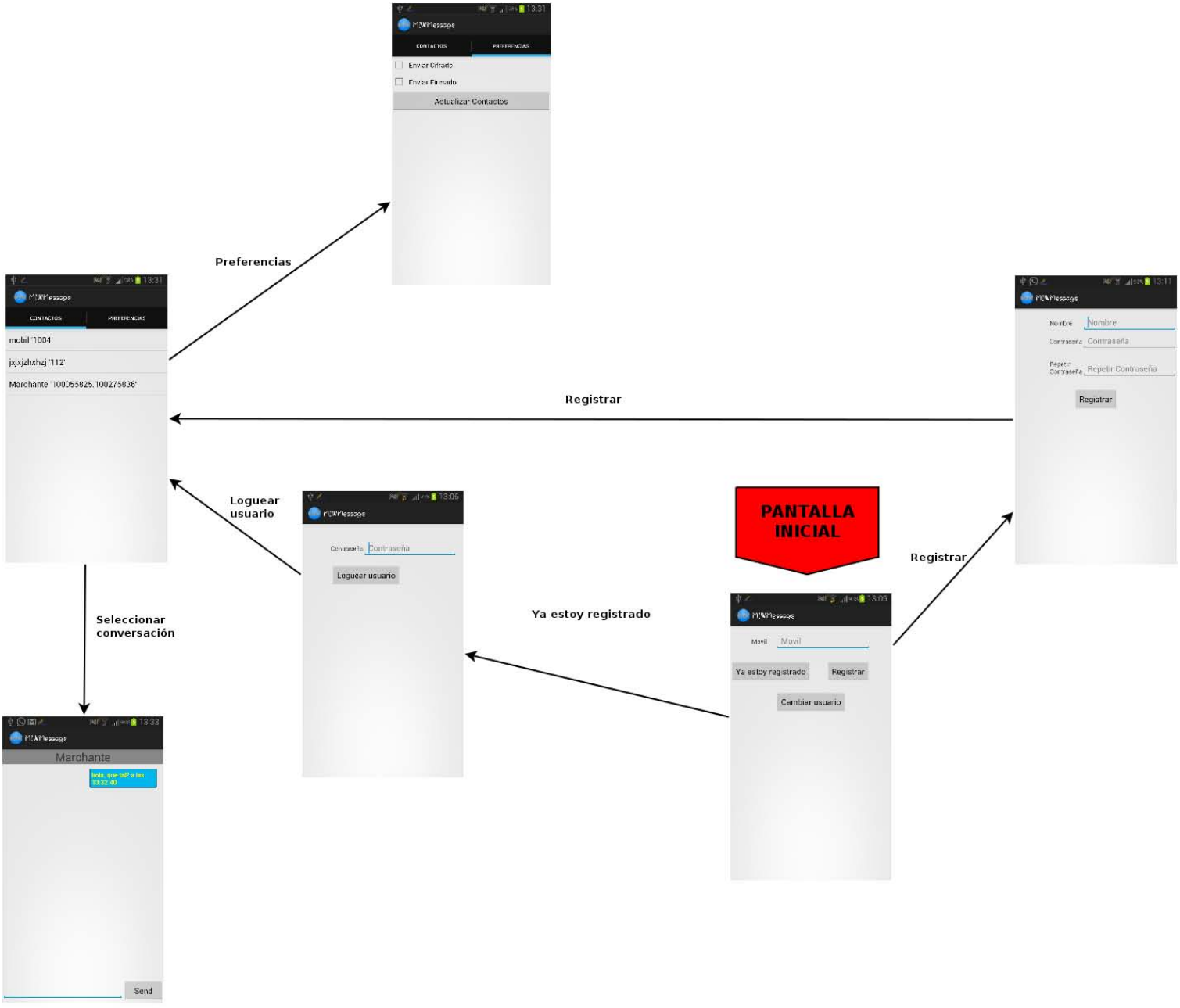

Figura 10: Mapa de Pantallas

Comunicación con el servidor

En este apartado, primeramente se detallarán los tipos de mensajes existentes y después se especificará el protocolo de comunicación entre la aplicación cliente y el servidor. Se mostrará también cada uno de los mensajes intercambiados entre ambos.

Tipos de mensajes

Dependiendo del tipo de mensaje existen diferentes contenidos posibles para un mensaje de comunicación. A continuación se especifican los distintos tipos de mensaje de la aplicación cliente con el servidor.

Tipo	Código	Descripción
SignUp	1	Utilizado para darse de alta en la aplicación
ContactRequest	2	Enviado para solicitar contactos de MIWMessage
ContactResponse	3	Respuesta a una petición de contactos.
ChatMessage	4	Mensaje de chat a otro usuario.
Connection	5	Indica que un usuario se ha conectado a MIWMessage.
Response	6	Mensaje de respuesta genérico.
Revocation	7	Revoca la clave pública de un usuario.
KeySend	8	Envío una clave pública de un contacto.
Upload	9	Sube una clave pública de un usuario al servidor.
KeyRequest	10	Petición de la clave pública de un contacto.

Tabla 37: Tipos de mensajes MIWMessage

Alta en el sistema

El alta en MIWMessage permite acceder a las funcionalidades de la aplicación. El procedimiento de alta en MIWMessage se describe a continuación:

1. El usuario establece su Alias de usuario y su número de móvil. La aplicación móvil envía un mensaje tipo "SignUp" al servidor de MIWMessage, indicando su teléfono móvil y su alias.
2. El servidor devuelve un mensaje de respuesta con código 201 al usuario si la cuenta de usuario se ha creado correctamente. El cuerpo de este mensaje incluye un token aleatorio generado por el servidor de 16 bytes. Este número, deberá ser utilizado como *sourceID* en el resto de mensajes enviados por el usuario.

Los mensajes intercambiados entre la aplicación Android y el servidor se muestran a continuación:

Figura 11: Proceso de alta

XML de envío:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>66666666</idSource>
      <idDestination>setichat@appspot.com</idDestination>
      <idMessage>2d46f3c49a2c6b7a2</idMessage>
      <type>1</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <signup>
        <nick>Alice</nick>
        <mobile>666666666</mobile>
      </signup>
   </content>
</message>
```

XML de respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>setichat@appspot.com</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>2d46f3c49a2c6b7a2</idMessage>
      <type>6</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <response>
        <responseCode>201</responseCode>
        <responseMessage>D009D2EEE117B5BD2DB2F8FF2DACF27D</responseMessage>
      </response>
   </content>
</message>
```

Petición de contactos

MIWMessage incluye un sistema que detecta automáticamente los contactos de la agenda de dicho usuario lo que resulta muy cómodo para encontrar los contactos registrados. Para ello, cada vez que se inicia la aplicación móvil, se realizará el siguiente procedimiento:

1. El cliente Android envía una petición de contactos al servidor de MIWMessage. Esta petición incluye todos los teléfonos móviles de la agenda del teléfono que no están ya incluidos en la lista de contactos.
2. El servidor de MIWMessage responde con los contactos solicitados que están dados de alta en el servicio MIWMessage. Por cada contacto se devuelve el número de teléfono móvil y el nick.

Figura 12: Proceso de petición de contactos

XML de envío:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>D009D2EEE117B5BD2DB2F8FF2DACF27D</idSource>
      <idDestination>setichat@appspot.com</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>2</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <mobileList>
        <mobile>111111111</mobile>
        <mobile>456165166</mobile>
        <mobile>156165165</mobile>
        <mobile>156161111</mobile>
```

```
        <mobile>165161656</mobile>
        <mobile>506506066</mobile>
        <mobile>000565050</mobile>
        <mobile>065065555</mobile>
        <mobile>498819111</mobile>
        <mobile>984616516</mobile>
        <mobile>661681818</mobile>
        <mobile>651616181</mobile>
        <mobile>050560005</mobile>
        <mobile>655166060</mobile>
        <mobile>466461611</mobile>
        <mobile>161616166</mobile>
      </mobileList>
   </content>
</message>
```

XML de respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>setichat@appspot.com</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>3</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <contactList>
        <contact>
          <mobile>679370720</mobile>
          <nick>Alice</nick>
        </contact>
        <contact>
          <mobile>645814714</mobile>
          <nick>Alicia</nick>
        </contact>
        <contact>
          <mobile>630746483</mobile>
          <nick>Bob</nick>
        </contact>
        <contact>
          <mobile>630746482</mobile>
          <nick>Benito</nick>
        </contact>
      </contactList>
   </content>
</message>
```

Conexión de usuario

La aplicación móvil de MIWMessage utiliza un servicio para recibir mensajes cuando la aplicación no está en primer plano. Aun así, los periodos en los que el teléfono se encuentra apagado, sin cobertura o sin el servicio iniciado pueden ocasionar la pérdida de mensajes. Para ello, cada vez que se inicia la aplicación móvil debe realizarse el siguiente procedimiento:

1. La aplicación móvil envía un mensaje de conexión al servidor.
2. El servidor envía un mensaje de respuesta con código 202.
3. El servidor envía cualquier mensaje pendiente de enviar al usuario recién conectado (si los hay).

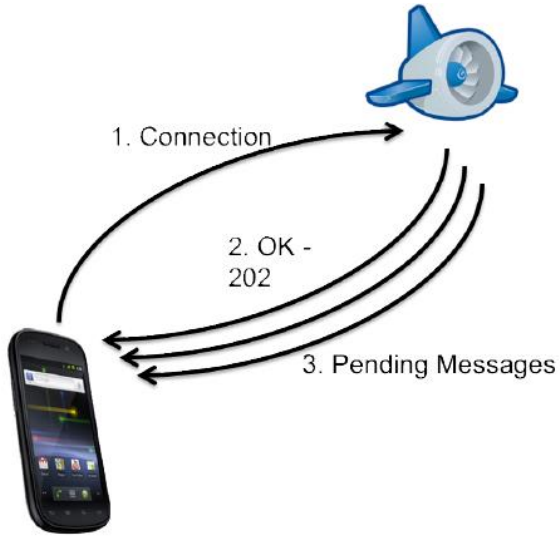

Figura 13: Proceso de notificación de conexión

XML de envío:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>D009D2EEE117B5BD2DB2F8FF2DACF27D</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>5</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <connection>
      </connection>
   </content>
</message>
```

XML de respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
```

```
      <idSource>setichat@appspot.com</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>6</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <response>
        <responseCode>202</responseCode>
        <responseMessage>User connected. Sending pending messages...</responseMessage>
      </response>
   </content>
</message>
```

Envío de un mensaje

El envío de mensajes de chat se realiza siempre a través del servidor. Los mensajes de chat pueden enviarse firmados y cifrados. Para cualquiera de los dos casos, será necesario disponer previamente de la clave correspondiente. El procedimiento a seguir para el envío de un mensaje de chat es el siguiente:

1. El usuario escribe el mensaje a enviar en un cuadro de texto y pulsa el botón enviar. La aplicación móvil encapsula el mensaje en un mensaje, que envía al servidor.
2. El servidor envía un mensaje de respuesta (con código 200) al emisor del mensaje. En caso de que el destinatario no existiese en el servidor, envía un mensaje de respuesta con código 408.
3. Si el destinatario del mensaje está dado de alta, el servidor envía el mensaje de chat al mismo. Si el usuario está dado de alta, pero no conectado, el mensaje es encolado.

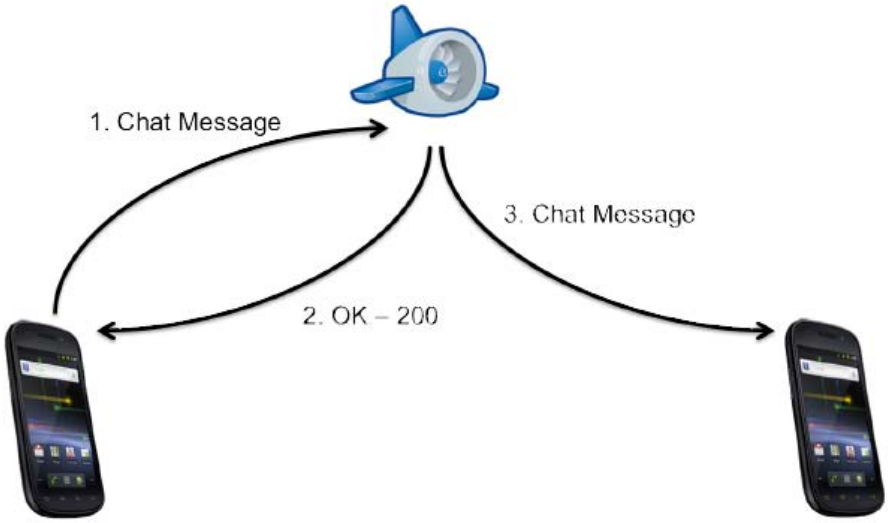

Figura 14: Proceso de envío de mensajes de chat

XML de envío:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>D009D2EEE117B5BD2DB2F8FF2DACF27D</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>4</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <chatMessage>hi Alice</chatMessage>
   </content>
</message>
```

XML de respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>setichat@appspot.com</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>6</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <response>
        <responseCode>200</responseCode>
        <responseMessage>
        Chat Message correctly received by SeTIChat server
        </responseMessage>
      </response>
   </content>
</message>
```

Descarga de claves

Los usuarios de MIWMessage pueden solicitar la descarga de claves para su uso en las tareas de cifrado y firma. Un usuario puede solicitar la clave pública de cualquier otro usuario. Para evitar la sobrecarga del sistema con claves que no van a ser utilizadas, sólo se solicitarán las claves de los usuarios con los que un usuario interactúa. El procedimiento de petición de clave es el siguiente:

1. La aplicación móvil solicita al servidor la clave de un usuario determinado (esta acción debe ser transparente al usuario de la aplicación).
2. El servidor responde con un mensaje que incluye la clave solicitada. En caso de que la clave solicitada no se encuentre disponible (el usuario no lo haya subido) se devolverá un mensaje de error con código 410.

Figura 15: Proceso de petición de clave pública

XML de envío:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>D009D2EEE117B5BD2DB2F8FF2DACF27D</idSource>
      <idDestination>setichat@appspot.com</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>10</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <keyrequest>
        <type>public</type>
        <mobile>679370721</mobile>
      </keyrequest>
   </content>
```

```
</message>
```

XML de respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>setichat@appspot.com</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>8</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <keyrequest>
        <type>public</type>
        <key>
        MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAOzP3QnbfZJJ6OHmldfvOu+aFlMI8Ge0ygm911SF3s
        7OJ6uUPWdm2utAKXtbhHITJ4csgQhuNvG9/qWkCLWEgK8CAwEAAQ==
        </key>
        <mobile>679370721</mobile>
      </keyrequest>
   </content>
</message>
```

Subida de clave pública

Los usuarios de MIWMessage pueden subir su clave pública en base64 al servidor. Subir la clave pública permite al usuario recibir mensajes cifrados y que el resto de usuarios puedan comprobar la firma de sus mensajes. Si un usuario envía una nueva clave al servidor teniendo ya una, la anterior será revocada. Para ello, se enviará un mensaje de revocación a todos los usuarios que tengan al usuario de la clave revocada como contacto. El proceso de subida de clave y revocación (en caso necesario) se muestra a continuación:

1. El dispositivo genera una clave. La clave es enviada al servidor.
2. El servidor envía un mensaje de respuesta con código 203 al emisor del mensaje.
3. En caso de que sea necesaria la revocación de la clave anterior, el servidor envía un mensaje a cada usuario que tenga como contacto al usuario que ha cargado el nuevo certificado.

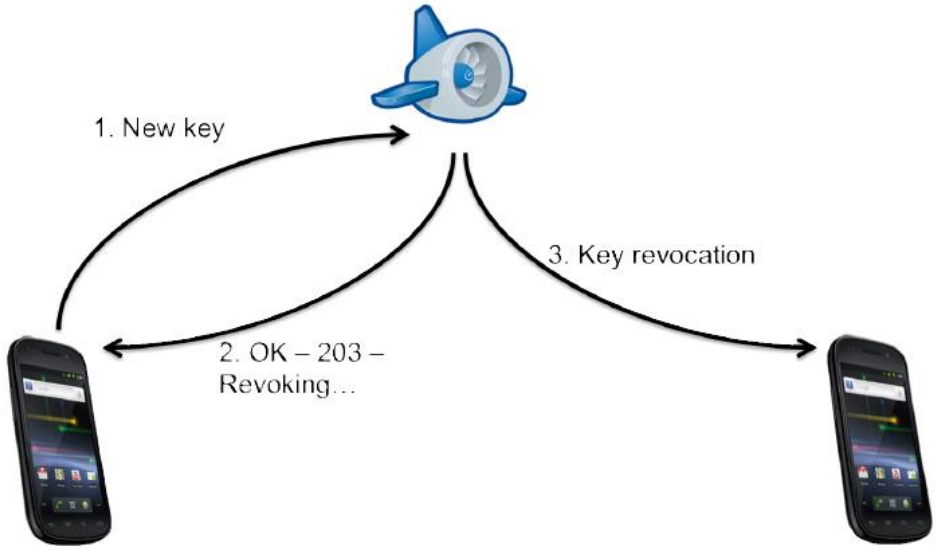

Figura 16: Proceso de subida de nueva clave (revocación de la antigua)

XML de respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>setichat@appspot.com</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>7</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <revokedMobile>679370721</revokedMobile>
   </content>
</message>
```

Seguridad

En este apartado se detallarán las estrategias criptográficas aplicadas en la aplicación. Se ha estructurado en tres apartados: seguridad de la base de datos local del cliente, seguridad en las comunicaciones y posibles vulnerabilidades del sistema implementado.

Seguridad de la base de datos local

Se ha realizado un estudio exhaustivo de la información sensible en la base de datos. Para evitar que un atacante pueda recuperar dicha información sensible si obtuviese la base de datos, se ha utilizado un sistema criptográfico conocido como ***Password Based Encryption (PBE)***. A continuación se muestra un esquema del sistema criptográfico para su mejor comprensión:

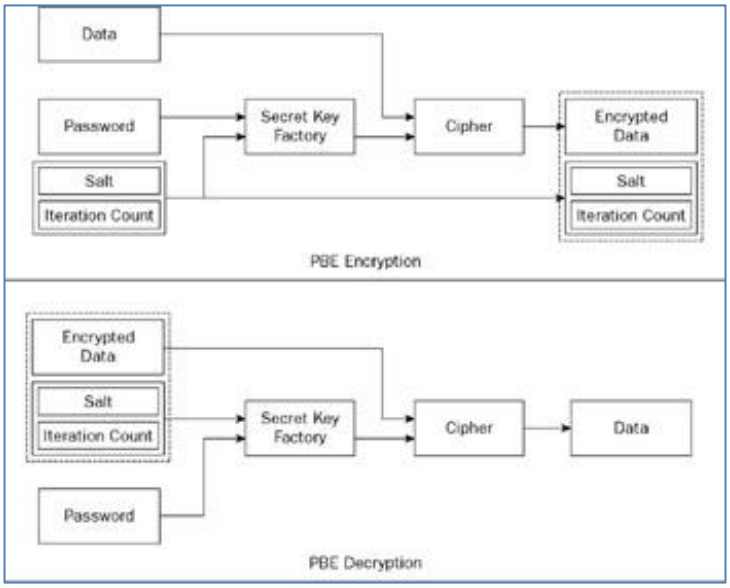

Figura 17: Esquema Password Based Encryption

Realizamos un cifrado de los datos sensibles de la base de datos por medio de Password Based Encryption [PBE], que es un modo seguro para guardar la contraseña en el que se genera una clave secreta con el password del usuario, un salt y un número de iteraciones. Si estos tres valores son siempre iguales, genera la misma clave secreta. Por tanto, esa clave secreta es la que se usará para cifrar toda la información sensible en la base de datos, incluido el password.

Para verificar el password se realizan cuatro pasos:

1. El usuario introduce la contraseña.
2. Se genera una clave secreta (simétrica) con esa contraseña, con el salt y con el mismo número de iteraciones con el que se generó la clave secreta la primera vez.
3. Se descifra el *password* cifrado de la base de datos con la clave secreta.
4. Se comprueba si el *password* descifrado de la base de datos es igual que el *password* introducido por el usuario para entrar.

En el caso de que el usuario entre en la aplicación, se guarda la clave secreta para cifrar/descifrar todos los datos necesarios de la base de datos.

En cuanto al salt, lo hemos elegido de 16 bytes ya que para ser seguro tiene que tener como mínimo 8 bytes y como se utiliza para cifrar AES-128, se ha definido un salt de 16 bytes. Otra opción era elegir un salt de 32 bytes y cifrar con AES-256 pero se estimó que para la relación rendimiento-seguridad era mejor 16 bytes.

Por otro lado, se ha generado un hash con 1000 iteraciones debido a que se ha estimado que es lo óptimo para que sea lo suficientemente segura y perjudique al rendimiento de la aplicación.

A continuación se describen cada uno de los campos de la base de datos y la justificación de por qué se ha decidido o no aplicar PBE que se ha considerado que contienen información sensible:

Atributos de la tabla **USUARIO:**

- ***Token***: se cifra porque si un atacante lo obtuviese podría suplantar a nuestro usuario en la aplicación puesto que para la autenticación con el servidor sólo hace falta enviarle nuestro token.

- **Nick**: no se cifra porque es algo conocido por todo el mundo que tenga nuestro número de móvil y no podrían realizarnos ningún ataque.

- **Mobile**: no se cifra porque sólo con el número de móvil no podrían atacarnos, como mucho enviarnos mensajes ofensivos, pero por motivos de rendimiento hemos decidido no cifrarlo puesto que no nos genera un gran impacto la pérdida de su confidencialidad.

- ***Password***: Se cifra porque si un atacante consiguiese nuestro terminal, podría acceder a éste con el número de móvil y el password realizando así nuestra suplantación en la aplicación.

- **Salt**: No se cifra porque es necesario saber el salt para poder crear la clave necesaria para validar la contraseña que introduce el usuario, con la contraseña cifrada que hay almacenada en la base de datos. Cada usuario tiene un salt diferente para darle más seguridad a la aplicación no sabiendo el salt de todos los usuarios obteniendo sólo el de uno. Para motivos de seguridad, se ha elegido utilizar un salt de 16 bytes, por lo que hay 2128 posibles valores de dicho salt, reduciendo así la posibilidad de descifrarlo mediante un ataque de fuerza bruta si no tienen acceso al contenido de la base de datos. Si este salt fuese conocido (junto con el número de móvil) por el atacante, podría realizar ataques de fuerza bruta y de diccionario para adivinar la contraseña de un usuario, por lo que también es responsabilidad de este introducir una contraseña segura para que dichos ataques tuviesen mucha más dificultad de realizarse.

- **Clave pública**: la clave pública del usuario no va cifrada porque lo puede obtener cualquier contacto que se lo solicite al servidor, por lo que al cifrarla sólo perderíamos rendimiento en la aplicación.

- ***Clave privada***: va cifrada en la base de datos porque es información única y exclusiva del usuario. La revelación de esta supondría que el atacante pudiese firmar mensajes por el usuario y leer todas las conversaciones enviadas al usuario con tan sólo hacer escuchas de los mensajes.

- **Fecha Claves**: la fecha de generación de las claves van guardadas en claro puesto que no supone una gran amenaza saber en qué fecha se generaron. Este dato es guardado puesto que la aplicación cada vez que está en un mes posterior al mes en que se generaron las fechas, vuelve a generar otro par de claves para el usuario revocando las anteriores para darle más seguridad. Si le roban la clave privada, sólo les servirá para el mes en que fue creada esa clave.

Atributos de la tabla **CONTACTO:**

- **Número Contacto**: es el número de móvil del contacto de un usuario. Esta información no se cifra puesto que los números de móvil son una información que no posee mucha confidencialidad, por lo que no nos serviría de mucho cifrarlo para esconderlo.

- **Nick**: es el Nick del contacto. Es enviado por el servidor en claro, y puede llegar a ser desde un nombre como una frase cualquiera, por lo que no va cifrado.

- **Clave pública**: la clave pública del contacto no va cifrada porque la puede obtener cualquier contacto que se lo solicite al servidor, por lo que al cifrarla sólo perderíamos rendimiento en la aplicación.

- **Contacto agenda**: identifica si el contacto es de la agenda, o ha abierto un chat con el usuario y pero no le tiene en la agenda. No va cifrado porque es una información irrelevante.

Atributos de la tabla **MENSAJE:**

- **Numero Contacto**: es el número de móvil del contacto de un usuario. Esta información no se cifra puesto que los números de móvil son una información que no posee mucha confidencialidad, por lo que no nos serviría de mucho cifrarlo para esconderlo.

- **Token**: se cifra porque si un atacante lo obtuviese podría suplantar a nuestro usuario en la aplicación puesto que para la autenticación con el servidor sólo hace falta enviarle nuestro token.

- **Mensaje_Chat**: se cifra porque se considera información privada de un usuario y aunque éste decidiese enviar dicha información sin cifrar, de cara a almacenar en base de datos, se cifra para mantener la confidencialidad de las conversaciones.

- **Recibido_Enviado**: que el mensaje haya sido recibido o enviado no es una información relevante y menos sabiendo que el contenido del mensaje está cifrado.

- **Cifrado**: que el mensaje esté cifrado es algo que hay que saber así que no tiene sentido que esté cifrado.

- **Id Mensaje Secuencial**: con el identificador incremental de los mensajes de chat no se podrá hacer ningún ataque, por lo que no es información relevante.

- **Hora Mensaje**: la hora del mensaje no es una información relevante y menos sabiendo que el contenido del mensaje está cifrado.

- **Fecha Mensaje**: la fecha del mensaje no es una información relevante y menos sabiendo que el contenido del mensaje está cifrado.

- **Firmado**: que el mensaje esté firmado es algo que hay que saber así que no tiene sentido que esté cifrado.

- **Verificado**: que el mensaje esté verificado no es un dato relevante, por lo que no va cifrado.

Seguridad en las comunicaciones

La seguridad en las comunicaciones se dividirá en dos secciones: protocolo utilizado en el canal de transmisión de los mensajes y las técnicas criptográficas utilizadas para proteger la confidencialidad y la integridad de los mensajes intercambiados entre los usuarios.

Para el canal de transmisión de los mensajes se utiliza el protocolo HTTPS.

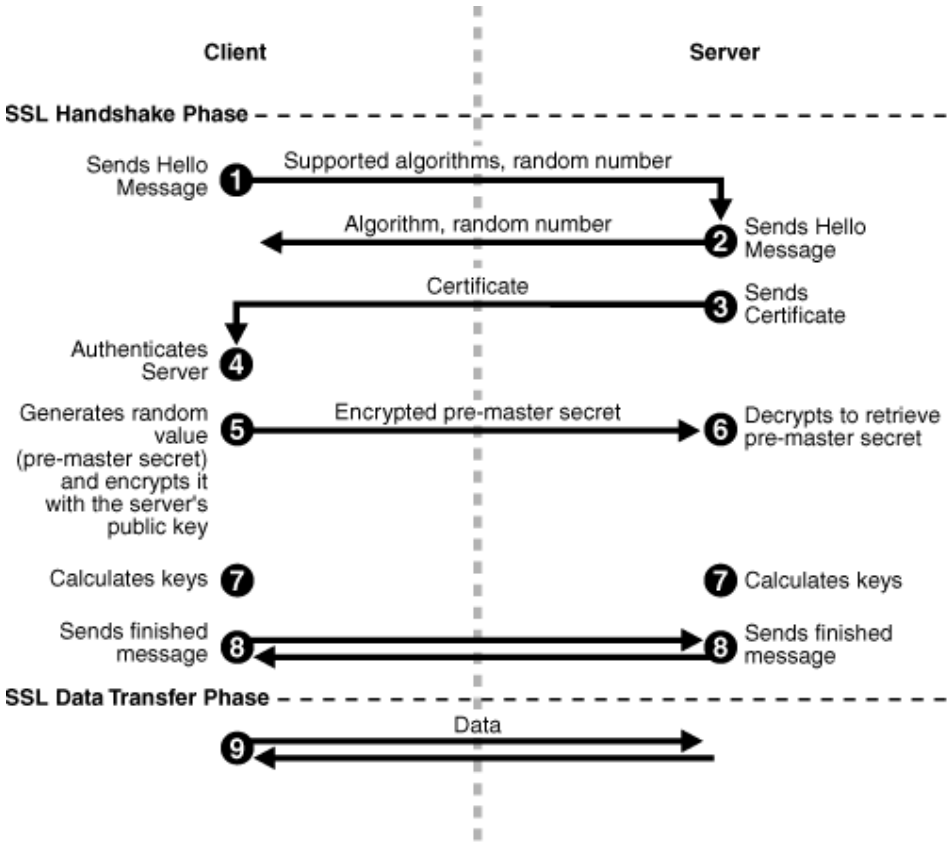

Figura 18: Protocolo HTTPS

Gracias a este protocolo se establece un canal seguro para evitar el ataque de Man in the middle entre MIWMessage y el servidor.

Para las técnicas criptográficas utilizadas para la protección de la confidencialidad y la integridad de los mensajes de texto intercambiados entre usuarios se han utilizado técnicas de cifrado y de firmado.
A continuación se expone el esquema general del cifrado de un mensaje de texto.

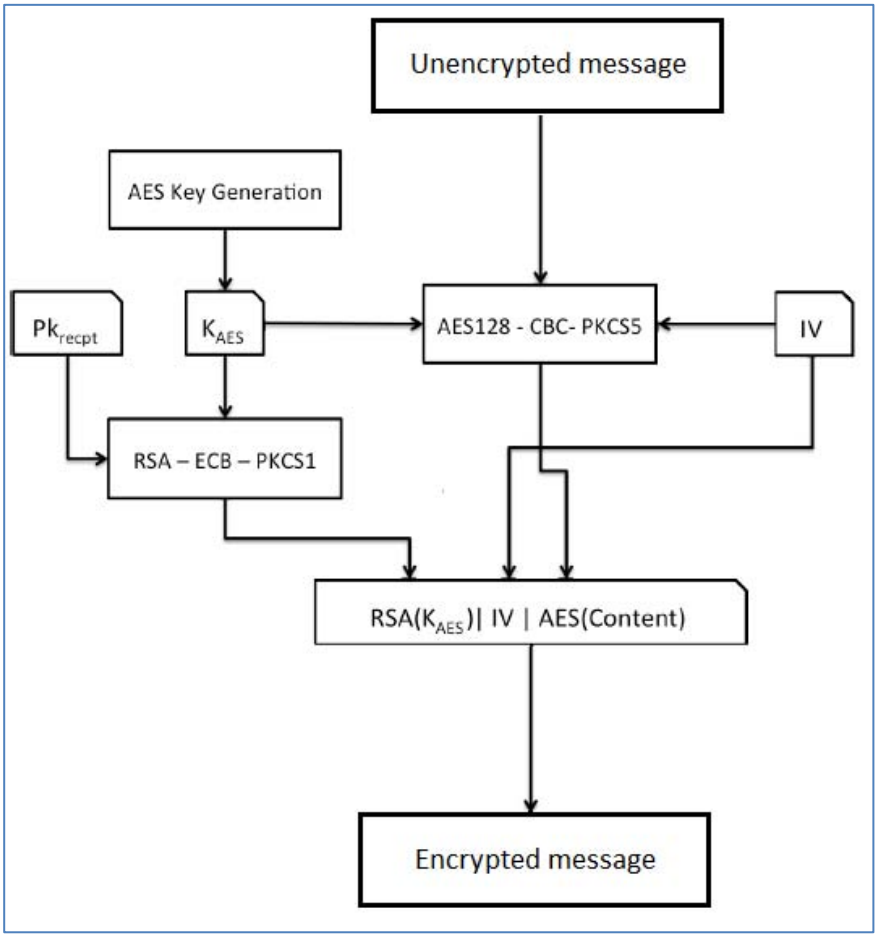

Figura 19: Esquema de cifrado de mensajes

Como se aprecia en el esquema:

1. Se genera una clave de 16 bytes (K_{AES}) y un vector de inicialización (IV).
2. Se aplica un cifrado de clave simétrica al mensaje con AES-128 en modo CBC utilizando la clave y el vector de inicialización generados anteriormente. El cifrado AES en modo CBC se realiza acorde al siguiente esquema.

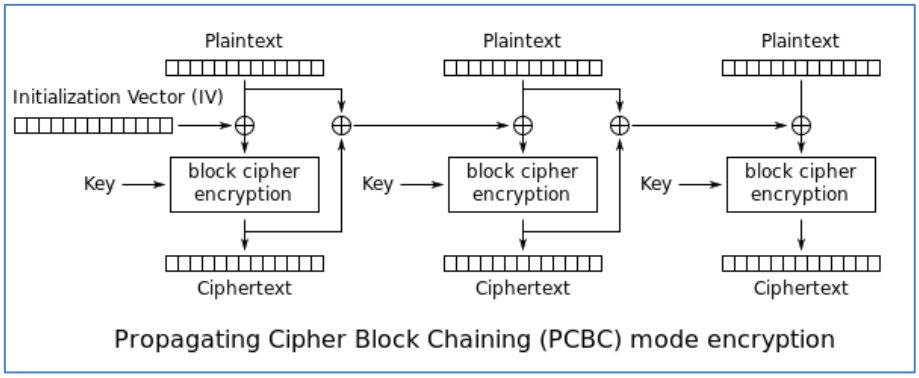

Figura 20: Esquema modo CBC

3. Se solicita la clave pública del contacto (Pk_{recpt}) al que será dirigido el mensaje (bien al servidor o bien a la base de datos si ya estaba solicitada).
4. Se aplica un cifrado de clave asimétrica sobre la clave simétrica (K_{AES}) con la que se ha cifrado el mensaje. Este cifrado se realiza con la clave pública del receptor (Pk_{recpt}) del mensaje por medio del cifrado RSA en modo ECB. A continuación se muestra el esquema del modo de cifrado ECB.

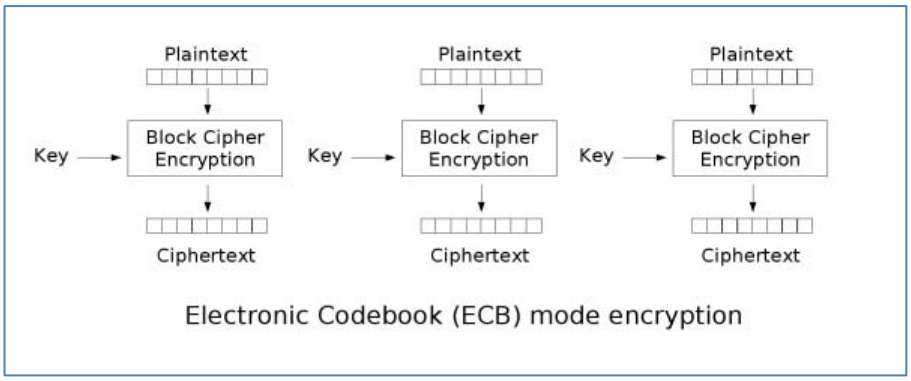

Figura 21: Esquema modo ECB

5. Se concatenan los bytes que se han obtenido con la clave simétrica cifrada ($RSA(K_{AES})$), el vector de inicialización en claro (IV) y los bytes del mensaje cifrado mediante AES (AES(Content)).
6. Con esto ya se tiene el mensaje cifrado.

A continuación se expone el proceso de descifrado de un mensaje:

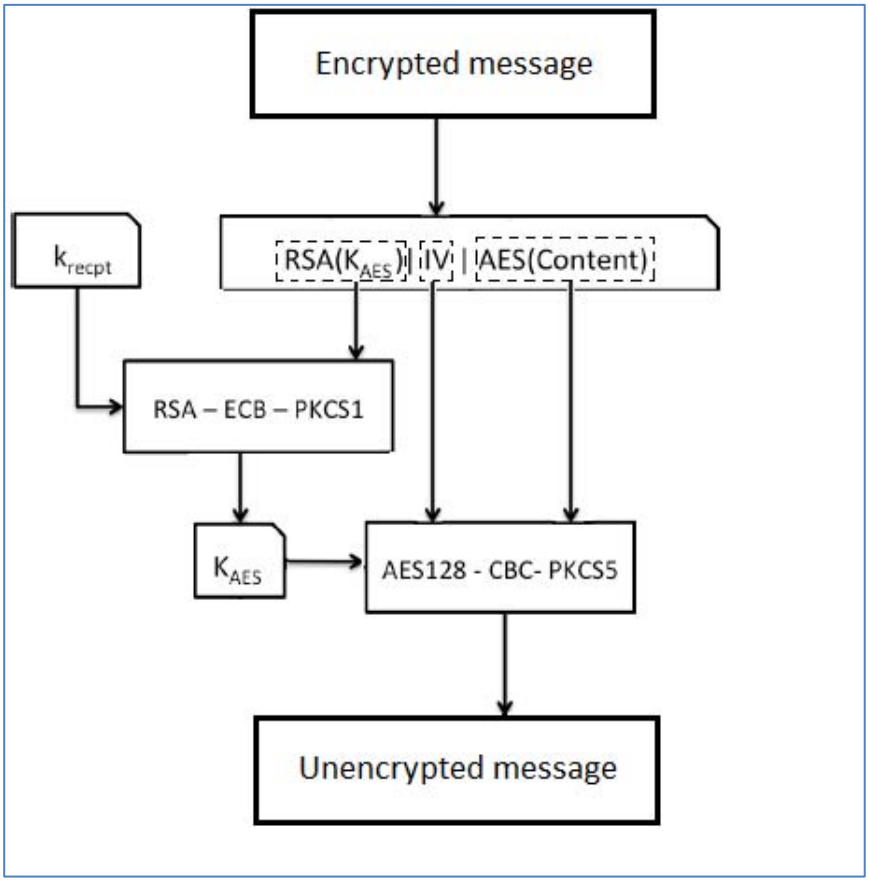

Figura 22: Esquema de descifrado de mensajes

1. El mensaje cifrado recibido se descompone en tres partes por número de bytes: 128 bytes que serán la clave simétrica cifrada con RSA (RSA(K_{AES})), 16 bytes que será el vector de inicialización (IV) y el resto de bytes que serán el mensaje recibido.
2. El receptor descifra la clave simétrica con su clave privada mediante RSA en modo CBC.
3. Se realiza un descifrado del mensaje a partir del vector de inicialización y la clave simétrica.
4. Con esto quedaría descifrado el mensaje.

A continuación se explica el proceso de firmado de mensajes:

1. El emisor aplica un algoritmo de hash SHA1 al menseje y lo cifra con su clave privada mediante RSA.
2. Añade al xml de envío una etiqueta llamada signature que contendrá la firma generada.
3. Con esto ya estaría firmado el mensaje en esta aplicación.

A continuación se explica el proceso de verificación de la firma:

1. El receptor aplica al mensaje en claro un algoritmo de hash SHA1.
2. Descifra la firma con la clave pública del emisor con RSA.
3. Compara el hash SHA1 del texto en claro con la firma descifrada.
4. Si son iguales quedaría verificada la firma.

En el caso del envío de mensajes cifrados y firmados, el orden de aplicación de los métodos criptográficos sería el siguiente:

1. Firmado.
2. Cifrado.

Este proceso hay que realizarlo en ese orden para que el emisor sea conocedor del mensaje que firma. Si fuese al contrario, el emisor firmaría un mensaje cifrado, por lo que podría firmar documentos maliciosos.

A continuación se muestran los xml que utiliza MIWMessage resultado de aplicar las técnicas expuestas anteriormente para un mismo mensaje.

XML con texto en claro:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>D009D2EEE117B5BD2DB2F8FF2DACF27D</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>4</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <chatMessage>hi Alice</chatMessage>
   </content>
</message>
```

XML con texto cifrado:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>D009D2EEE117B5BD2DB2F8FF2DACF27D</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>4</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <chatMessage>
      CI8x1WhWYGTqCQufauMQzeS12QpUfChGgA9J0MG3JRgIcQVb9nWOI9wBUxd8sf4Mdc31Xw0RBwZ
      FYMC5gn/cPwX79YSCQ3vYStEAIL6a7SJqyLS2R5XtC4x0FDcle1G90qjwKB38EisyBsZ/cariHq
      1OY+Xce314kybpwIQk3idaPXvfzSfJ5f6FX7jK5WLAAb/q2VSrBQm4+NyCCNipkw==
      </chatMessage>
   </content>
</message>
```

XML con texto firmado:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
```

```
   <header>
      <idSource>D009D2EEE117B5BD2DB2F8FF2DACF27D</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>4</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <chatMessage>hi Alice</chatMessage>
   </content>
   <signature>
   m71f8Lr5oZbz0PRf1rRB96chD4OTEr5pJwUoiusAiq1AbFG+6K18pbxty7QwSI4r3Xg/8gM4BPYp0/
   zI6UbnMUfs4vOfMihdxuce2/56aa1D16iuoTOomLA7glRyMUv1bYnusM+FjxkXdkjJoWuAi62Y+zrk
   kORrtBdKPZucNmE=
   </signature>
</message>
```

XML con texto cifrado y firmado:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <header>
      <idSource>D009D2EEE117B5BD2DB2F8FF2DACF27D</idSource>
      <idDestination>666666666</idDestination>
      <idMessage>vD1zfnZCK9QCdr4s9U0dfg==</idMessage>
      <type>4</type>
      <encrypted>false</encrypted>
      <signed>false</signed>
   </header>
   <content>
      <chatMessage>
      CI8x1WhWYGTqCQufauMQzeS12QpUfChGgA9J0MG3JRgIcQVb9nWOI9wBUxd8sf4Mdc31Xw0RBwZ
      FYMC5gn/cPwX79YSCQ3vYStEAIL6a7SJqyLS2R5XtC4x0FDcle1G90qjwKB38EisyBsZ/cariHq
      1OY+Xce314kybpwIQk3idaPXvfzSfJ5f6FX7jK5WLAAb/q2VSrBQm4+NyCCNipkw==
      </chatMessage>
   </content>
   <signature>
   m71f8Lr5oZbz0PRf1rRB96chD4OTEr5pJwUoiusAiq1AbFG+6K18pbxty7QwSI4r3Xg/8gM4BPYp0/
   zI6UbnMUfs4vOfMihdxuce2/56aa1D16iuoTOomLA7glRyMUv1bYnusM+FjxkXdkjJoWuAi62Y+zrk
   kORrtBdKPZucNmE=
   </signature>
</message>
```

Vulnerabilidades del sistema implementado

En este apartado nos centraremos en indicar la manera de solventar las posibles amenazas del sistema implementado, así como la forma de solucionarlas.

En un primer lugar, MIWMessage soluciona un posible ataque de hombre en el medio utilizando el protocolo SSL para el intercambio de mensajes. Esto subsanaría el problema, pero si un atacante encontrase la manera de realizar un ataque de falsificación de IP a la aplicación podría hacer que los mensajes enviados al servidor alojado en la dirección https://setichat.appspot.com fuesen enviados a la dirección IP de un servidor que fuese del atacante, suplantando de tal manera al servidor. Por lo que si los mensajes no fuesen cifrados, la información sensible que viaja por el canal sería transparente para el atacante, teniendo la posibilidad de leerlos; y si no fuesen firmados, podría modificarlos.

MIWMessage implementa un idMessage que sirve tanto a los clientes móviles como al servidor para identificar la respuesta de los mensajes que solicitan. Para más seguridad, se podría implementar una estructura que almacenase todos los idMessage en el cliente y en el servidor con la finalidad de que no se repitan nunca y si le llegase algún idMessage repetido descubriría que está recibiendo envíos de otro servidor. De esta manera, un cliente y un servidor podrían intercambiarse 16384 mensajes (128^2), puesto que el idMessage se compone de 16 bytes que son 128 bits. A pesar de que este idMessage tiene 16 bytes, al igual que la clave simétrica utilizada para el cifrado de los mensajes, jamás podríamos utilizarlo para tal fin puesto que el idMessage viaja en claro por la red, con lo que cualquier atacante podría descifrar nuestros mensajes captando el paquete, sin embargo, sí podría ser utilizado como vector de inicialización puesto que éste también viaja en claro.

MIWMessage tiene dos problemas a la hora de la autenticación:

- El servidor no se autentica ante el cliente.
- Un cliente móvil puede suplantar la identidad de otros usuarios.

Para solucionar el primer problema, podríamos implementar una autenticación mutua entre cliente y servidor mediante reto-respuesta en la que cuando el cliente solicite la conexión el servidor le envíe un array de bytes. El cliente lo firmaría y devolvería al servidor la firma de los datos y su certificado de clave pública para que pudiese verificarlo. Una vez autenticado el cliente, se realizaría el mismo proceso en orden inverso para que el servidor se autenticase contra el cliente.
Para más seguridad, si se autenticase de esta manera, se podría optar por la generación de dos pares de claves de las cuales un par se usaría para firmar autenticaciones y otro para firmar mensajes (como los DNI electrónicos) para evitar que si es un servidor malicioso, le envíe un reto que sea un documento en el que se compromete a cualquier tipo de cargo y éste lo firme y lo devuelva.

Para solventar el segundo problema, debido a que la aplicación almacena números de teléfonos móviles, el servidor podría enviar un mensaje de texto al teléfono con la contraseña la cual le solicitase antes de darle de alta en la aplicación para comprobar si realmente es ese usuario.

En contra de los inconvenientes respecto a la seguridad en la autenticación, el servidor guarda el token de forma interna cuando un usuario es registrado y nunca se mostrará al usuario evitando así que dicho token se publique y pueda usarse de forma malintencionada para la suplantación de la identidad de un usuario ya registrado. Esta ventaja del servidor se ve contrarrestada por el cliente puesto que en todos los mensajes que le envía al servidor, envía su token en claro en vez de mandar su número de móvil, en el campo idSource.

También la aplicación evita la suplantación de cualquier usuario por medio de otras personas en el caso de que le robasen el terminal o lo perdiese. Esta funcionalidad se solventa gracias a que la aplicación solicita la contraseña al usuario cada vez que accede a la aplicación.

Se soluciona el problema de que terceros puedan obtener información, de los mensajes que se envían, debido a que se puede realizar un envío del contenido de los mensajes cifrados.

El problema de que un servidor malicioso pudiese modificar los mensajes enviados por un usuario sin que el receptor se diese cuenta se soluciona realizando la firma. De esta manera, si el servidor modificase algún campo crítico del mensaje, el receptor sabría que no es el mismo mensaje generado por el emisor al verificar dicha firma.

También se ha solucionado el problema de la confidencialidad de la base de datos guardando los datos sensibles de la aplicación de forma segura mediante un cifrado realizado con *Password Based Encryption*. De esta forma, los datos catalogados como sensibles ya no están en la base de datos de forma legible.

Implementación

Esta fase está destinada a conocer las características del proyecto desarrollado, identificando la arquitectura del proyecto. La primera parte de este apartado se va a dedicar a explicar la arquitectura MVC (Modelo-Vista-Controlador) que se ha aplicado en este proyecto así como la justificación de sus componentes. La segunda parte está destinada a identificar las características de la arquitectura en el proyecto desarrollado.

Arquitectura MVC

El patrón de arquitectura MVC [MVC] define la organización independiente del **Modelo** (objetos de negocio), la **Vista** (interfaz de interacción con el usuario u otro sistema) y el **Controlador** (que define el flujo de trabajo de la aplicación). De esta manera, podemos subdividir el sistema en tres capas, consiguiendo tratar, de manera independiente: los datos y la lógica de negocio, la interfaz y el módulo de gestión de eventos y comunicaciones. De esta forma se consigue facilitar la tarea de desarrollar aplicaciones y su posterior mantenimiento.

A continuación se explicarán los tres componentes principales de este modelo:

- **Modelo:** constituye la representación de la información con la que opera el sistema y gestiona todos los accesos a dicha información implementando también los privilegios de acceso descritas en la lógica de negocio. El Modelo envía a la Vista la información que solicita para mostrar. Las peticiones de acceso o manipulación de información llegan al Modelo a través del Controlador.

- **Vista:** presenta el Modelo en un formato adecuado para interactuar con él, es decir la interfaz de usuario.

- **Controlador:** es el encargado de responder a los eventos (normalmente interacciones del usuario con el interfaz) e invocar peticiones al modelo cuando se realiza alguna solicitud de información. El controlador funciona como intermediario o *middleware* entre la Vista y el Modelo.

Estas son las relaciones entre una vista, un modelo y un controlador:

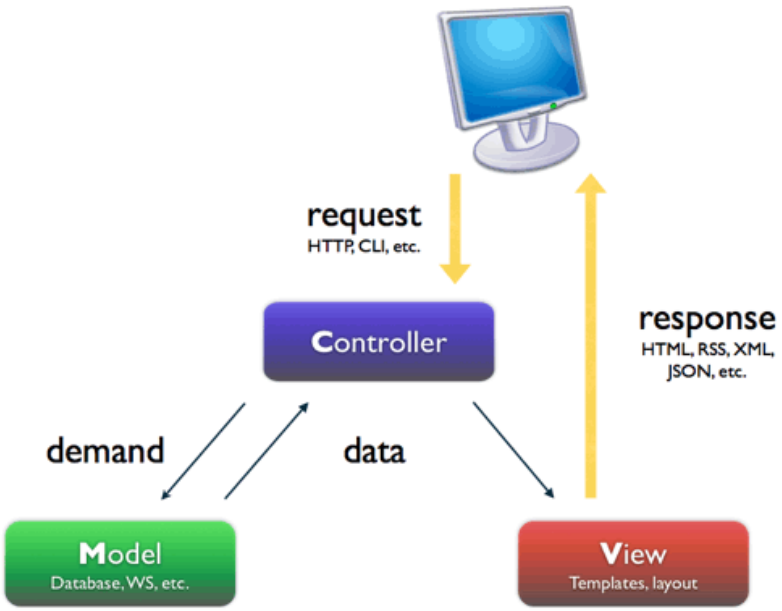

Figura 23: Esquema de arquitectura MVC

Esta arquitectura presenta grandes de benefícios:

- Permite facilitar la conversión del código. El código está muy encapsulado y actúa de manera casi independiente. Esto permite que su aprendizaje sea más rápido y reutilizar el código para otras aplicaciones.

- Vistas actualizadas: el programador no debe ocuparse de solicitar que las vistas actualicen, ya que este proceso se realiza automáticamente por el modelo de la aplicación.

- Cualquier modificación sujeta al dominio implica una modificación sobre el modelo y las interfaces del mismo, es decir, no es necesario modificar los mecanismos de comunicación y actualización de modelos contemplados en el controlador.

- Las modificaciones de la lista no afectará al modelo, simplemente se modifica la representación de información, no su tratamiento.

Las aplicaciones desarrolladas bajo esta arquitectura presentan una extensa flexibilidad y una mantenibilidad únicas comparadas con aplicaciones basadas en otros patrones.

Arquitectura cliente-servidor

Este proyecto se desarrolla bajo la arquitectura cliente servidor [ACS], que consiste en un modelo de aplicación distribuida en la que las tareas se reparten entre los proveedores de recursos o servicios. De esta forma, se centraliza el control de accesos, recursos e integridad; recursos que poseen mucha relevancia para una aplicación de mensajería. Este tipo de arquitectura se fundamenta en base al esquema que se presenta a continuación:

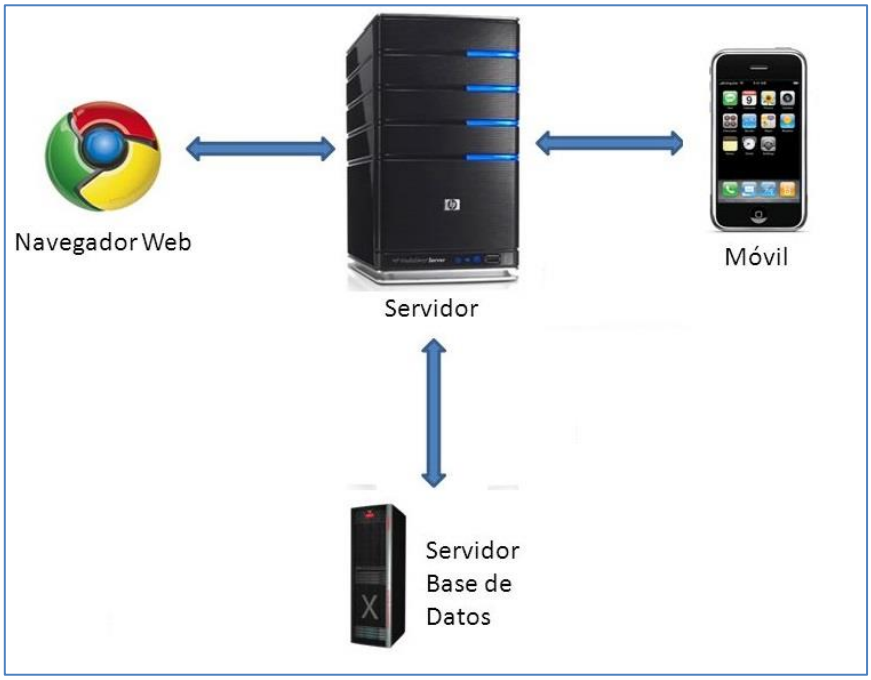

Figura 24: Arquitectura cliente-servidor

En este caso la parte del servidor se encarga de proveer del canal de mensajería a los clientes móviles así como los servicios necesarios para la autenticación.

4 Evaluación

Este apartado de evaluación tiene como objetivo demostrar la validez de la solución elaborada. Para que la solución se considere válida, debe satisfacer los objetivos definidos en la sección 1.1 Objetivos.

En este apartado se evalúa si el sistema desarrollado es válido a través de una serie de pruebas de validación. En primer lugar se presentará el proceso de evaluación llevado a cabo para validar el sistema. Más adelante se estudiarán diferentes casos de prueba y una vez definido esto, se procederá a extraer conclusiones obtenidas en este proceso de evaluación.

4.1 Proceso de evaluación

Este apartado corresponde a la primera parte de la evaluación del sistema. Este se compone a su vez de los sub-apartados: en el primero se escoge y diseña el método a evaluar, y el segundo se ejecuta dicho método.

Plan de pruebas

En este apartado se expone el plan de pruebas utilizado para la evaluación de este sistema. Lo primero que se va a realizar son los diferentes casos de prueba que permitan verificar que el sistema cumple con los requisitos establecidos en la fase de análisis. Más adelante, analizaremos los resultados obtenidos al verificar los casos de prueba.
En este caso, los casos de prueba están relacionados con los casos de uso definidos en el apartado de Casos de Uso presentados anteriormente. Debido a que todos los requisitos están contenidos en los casos de uso, si generamos los casos de prueba en función de los casos de uso podemos verificar en los requisitos cumplen el sistema que se ha desarrollado.

Definiremos los casos de prueba según esta plantilla:

CP-XX	
Nombre	
Descripción	
Caso de uso	
Precondición	
Secuencia	
Verificación	

Figura 25: Plantilla de Casos de Prueba

Casos de prueba

En este apartado, se va a proceder a presentar los casos de prueba siguiendo la plantilla del apartado anterior. Como ya mencionamos anteriormente, estos casos de pruebas tan basados en los casos de uso definidos en la fase de Análisis del sistema. Estos escenarios se van a evaluar en el sistema y se va a verificar el resultado de los mismos.

A continuación se presentan los casos de prueba necesarios para evaluar la aplicación:

CP-01	
Nombre	Acceder al sistema
Descripción	Se le permitirá al usuario registrarse en el sistema
Caso de uso	CU-01
Precondición	El usuario debe estar registrado en la aplicación
Secuencia	1. El usuario indica que ya está registrado en su terminal. 2. El sistema pide que introduzca la contraseña. 3. El usuario accede a la aplicación.
Verificación	Se presenta la Pantalla Contactos

Tabla 38: Caso de prueba: CP-01

CP-02	
Nombre	Registrar en el sistema
Descripción	El usuario desea registrarse en el sistema
Caso de uso	CU-02
Precondición	-
Secuencia	1. El usuario introduce el número a registrar. 2. El usuario indica que quiere registrarse. 3. El usuario introduce el resto de datos de registro. 4. El usuario se registra en la aplicación.
Verificación	Se presenta la Pantalla Contactos

Tabla 39: Caso de prueba: CP-02

CP-03	
Nombre	Cambiar usuario
Descripción	Se desea cambiar de usuario en la aplicación
Caso de uso	CU-03
Precondición	El usuario debe tener una cuenta en la aplicación
Secuencia	1. El usuario introduce el número de usuario al que desea cambiar. 2. El usuario indica que quiere cambiar de usuario. 3. El usuario introduce la contraseña. 4. Se cambia el usuario en la aplicación.
Verificación	Se muestra la Pantalla Contactos

Tabla 40: Caso de prueba: CP-03

CP-04	
Nombre	Enviar mensajes en claro
Descripción	El usuario desea enviar un mensaje en claro
Caso de uso	CU-04
Precondición	1. Tener deshabilitadas las opciones criptográficas 2. El usuario debe haber iniciado una conversación con un contacto.
Secuencia	1. El usuario introduce el mensaje que desea enviar. 2. El usuario indica que quiere enviar el mensaje.
Verificación	Se añade una burbuja de conversación con el mensaje enviado.

Tabla 41: Caso de prueba: CP-04

CP-05	
Nombre	Enviar mensajes cifrados
Descripción	El usuario desea enviar un mensaje cifrado
Caso de uso	CU-05
Precondición	1. Habilitar la opción de cifrado. 2. El usuario debe haber iniciado una conversación con un contacto.
Secuencia	1. El usuario introduce el mensaje que desea enviar. 2. El usuario indica que quiere enviar el mensaje.
Verificación	Se añade una burbuja de conversación con el mensaje enviado y un círculo rojo.

Tabla 42: Caso de prueba: CP-05

CP-06	
Nombre	Enviar mensajes firmados
Descripción	El usuario desea enviar un mensaje firmado
Caso de uso	CU-06
Precondición	1. Habilitar la opción de firmado. 2. El usuario debe haber iniciado una conversación con un contacto.
Secuencia	1. El usuario introduce el mensaje que desea enviar. 2. El usuario indica que quiere enviar el mensaje.
Verificación	Se añade una burbuja de conversación con el mensaje enviado y un círculo gris.

Tabla 43: Caso de prueba: CP-06

4.2 Análisis de resultados

En este apartado se analizan los resultados obtenidos en las pruebas descritas el apartado anterior. En el análisis de los resultados se va formalizar por medio de una tabla en la que se recojan los resultados obtenidos en el proceso de evaluación. Para cada prueba asociada se especificará la fecha en la que se realiza y el resultado de la prueba. En caso de no tener un resultado satisfactorio, será necesario añadir una descripción especificando los resultados que se obtienen.
A continuación se muestran los resultados obtenidos en el proceso de evaluación de los casos de prueba:

Caso de prueba	Fecha	Resultado	Descripción
CP-01	27/06/2015	Falla	
CP-02	27/06/2015	Falla	
CP-03	27/06/2015	Falla	
CP-04	27/06/2015	Falla	
CP-05	28/06/2015	Falla	
CP-06	28/06/2015	Falla	

Tabla 44: Evaluación de los Casos de Prueba

Como se puede observar en los resultados obtenidos, todos los casos de uso han sido verificados en el proceso de evaluación, por lo que podemos concluir que **el equipo de pruebas ha fracasado.**

5 Conclusiones

En esta sección se resumen las principales aportaciones del proyecto, se enumeran los problemas que se plantean a lo largo de su desarrollo y se discute la posible labor futura basada en el desarrollo actual y opiniones personales del trabajo.

5.1 Contribuciones

La principal aportación de este proyecto es el desarrollo de una aplicación cliente-servidor que permite el intercambio de mensajes entre clientes Android, con el fin de permitir una vía de comunicación que permita elegir los métodos criptográficos que quieren aplicar a un mensaje. Esta aplicación es muy útil desde el punto de vista didáctico debido a que se puede seleccionar los diferentes métodos y realizar técnicas de *sniffing* para capturar paquetes y ver como viaja la información por el canal. En consecuencia, este proyecto simplifica las versiones de mensajería modernas para darle un enfoque desde el punto de vista de la seguridad en su forma más pura y marca las directrices a seguir para ampliar la funcionalidad del propio sistema pudiendo realizar, una aplicación de mensajería más compleja y completa, con poco esfuerzo.

5.2 Trabajos futuros

En esta sección se enumerarán alguno de los trabajos futuros que podrán derivarse a partir de la realización de este proyecto.

Los primeros pasos a seguir deben estar orientados a desarrollar la parte de comunicación del lado del servidor, estableciendo así un sistema de transmisión de información y almacenaje centralizando el negocio de la aplicación.

El segundo paso podría ser transferir imágenes dentro de la propia aplicación como cadenas de bits, completando así una funcionalidad que hoy en día se considera básica en las aplicaciones de mensajería modernos. Analizando las protecciones que serían deseables para este tipo de mensaje.

Otro paso interesante, sería el de almacenar las sesiones en el servidor. Con esta medida, no se perderían las sesiones en caso de que se desinstalara la aplicación del terminar ya que estas almacenan los elementos de sesión en local y si esta información se pierde, el usuario pasa a ser inservible. Exactamente igual que pasa con los mensajes; sería deseable que las conversaciones se

almacenaran también en el servidor para no perder la información que tanto trabajo ha costado proteger.

5.3 Problemas encontrados

El principal problema ha sido la parte del servidor. En un primer momento se estableció un proyecto que incluía tanto la parte de servidor como la parte de cliente. Por motivos de tiempo fruto de un planteamiento inicial demasiado ambicioso, tomó la determinación de analizar las técnicas criptográficas en profundidad para conseguir una aplicación que pudiese desenvolverse, en motivos de seguridad, en un entorno real; haciendo que este hecho prevaleciese ante el hecho de desarrollar la parte completa del servidor y realizar la documentación pertinente. Se comenzó utilizando un servidor que implementaba un Channel API en Google Engine y al final decidimos dejar el servidor y centrar el grueso del trabajo en la parte de la seguridad de la aplicación.

Referente al desarrollo de la aplicación móvil, los problemas principales surgen principalmente a raíz de la experiencia en tecnología utilizada. Android cuenta con bastante similitud con lenguajes de la familia Java, pero no deja de ser un lenguaje orientado a tecnologías móviles, por lo que cuenta con muchas peculiaridades y limitaciones impresas por los dispositivos en las que se ejecutan Esto supuso una desventaja debido a que tuvimos que tomarnos un tiempo considerable para aprender el lenguaje y desenvolvernos con soltura.

Un apartado que nos ha llevado más tiempo del que se estipulo en un principio fue el análisis de las necesidades criptográficas de la aplicación. Se tuvo que analizar cómo proteger los datos que se almacenan en local y la forma de proteger los mensajes que se transmiten. Bien es cierto que en un principio se podría pensar que sencillamente hay que realizar un pequeño análisis de las posibilidades y decantarse por una tecnología; pero en la práctica esta tarea ha sido mucho más compleja. Ha sido necesario hacer un análisis de eficiencia, a que al ser una aplicación de comunicación, el tiempo de envio-recepción de los mensajes no es algo que se pueda descuidar; por lo que no es una cuestión de mera eficacia.
A la hora de realizar los cifrados, se han evaluado las técnicas que menos podrían comprometer la eficiencia de la aplicación, ya que no tendría sentido que al receptor le llegase el mensaje 3 minutos después de que el emisor lo hubiese enviado. Por este motivo, por ejemplo se cifra con un sistema de clave asimétrica la clave simétrica con la que se descifrará el contenido del mensaje, dado que cifrar el mensaje directamente con un sistema de clave asimétrica habría retrasado notablemente el proceso de comunicación.

5.4 Opinión personal

La realización de este proyecto ha sido un gran desafío. En particular, como se discutió en la sección anterior, ha habido muchos problemas en la implementación de la aplicación por la falta de experiencia en la tecnología de desarrollo. Al concluir esta sección, nos gustaría destacar algunas de las habilidades que se han obtenido, a lo largo del Master en Ingeniería Web y desarrollando este TFM.

A lo largo del master, se han adquirido muchas habilidades que se han puesto en práctica y han sido muy útiles para hacer este proyecto. Una de estas habilidades es la capacidad de generar programar escribiendo código mejor estructurado y mucho más limpio, utilizando los principios del paradigma orientado a objetos así como patrones de diseño y sobre todo refactoring (TDD). Se han ampliado notablemente los conocimientos en la plataforma Android, de la que adquirimos las pinceladas básicas en la asignatura “Desarrollo de Aplicaciones para Sistemas Moviles” (DASM). Un pilar básico de este desarrollo ha sido la comunicación con la parte de servidores que se estudió en la asignatura “Desarrollo de Aplicaciones Web Distribuidas de Código Abierto” (JEE).

Otra asignatura que ha demostrado ser muy útil para el desarrollo de la documentación del proyecto, ha sido "Metodologías Pesadas para Desarrollo Web"(MPDW), en la que se explicó la importancia de los “artefactos” presentes en un proyecto y la importancia de que estén correctamente estructurados y escritos.

Para concluir, el pilar básico para la realización de este TFM ha sido la asignatura “Seguridad en la Programación Web” (SPW) de la cual se ha extraído el conocimiento necesario para desarrollar la parte fundamental de la aplicación.

6 Bibliografía

Referencias utilizadas para la realización del proyecto.

[BTC] "Bases y tipos de cotización contingencias comunes" Ministerio de Empleo y Seguridad Social. 2014. Disponible [Internet]:
http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm

[CAS] "Modelo en Cascada" Ing. Jymmy Guevara. 2012. Disponible [Internet]:
https://sites.google.com/site/adai6jfm/modelo-cascada

[VOL] Volere, "Plantilla de Especificación de Requisitos".2006. Disponible [Internet]:
http://www.volere.co.uk/pdf%20files/template_es.pdf

[MVC] Deacon, J. (2009). "Model-view-controller (mvc) architecture" .2006. Disponible [Internet]:
http://www.jdl.co.uk/briefings/MVC.pdf

[ACS] Berson, A. (1992). Client-server architecture (No. IEEE-802). McGraw-Hill.

[PBE] Bellovin, S. M., & Merritt, M. (1992, May). Encrypted key exchange: Password-based protocols secure against dictionary attacks. In Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on (pp. 72-84). IEEE.
http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=213269&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D21326

Control de versiones

Versión	Fecha de finalización	Descripción
1.0	10/06/2015	Redacción del apartado de la introducción y el esqueleto general de la memoria.
2.0	11/06/2015	Redacción del apartado de Gestión de proyectos Software.
3.0	12/06/2015	Redacción del apartado 3.1 Análisis, Diseño e Implementación
4.0	22/06/2015	Redacción del apartado 4 Evaluación
5.0	24/05/2015	Redacción del resto del apartado Conclusiones
6.0	27/06/2014	Revisión de los apartados 1 y 6. Glosario de términos. Inserción de títulos y corrección de estilos
7.0	30/06/2014	Revisión y correcciones de los apartados en inglés. Maquetación Final

Tabla 45: Control de versiones